Memoria Práctica 5
Kubernetes y Raft
Sistemas Distribuidos

Héctor Lacueva Sacristán NIP: 869637 Adrián Nasarre Sánchez NIP: 869561

GRUPO TARDES 3-3

Fecha: 20/06/2025

Índice

Introducción

En esta memoria se describe el desarrollo de la Práctica 5 de Sistemas Distribuidos, que consiste poner en funcionamiento un servicio replicado basado en Raft.

El objetivo principal de esta práctica es el despliegue de un servicio de almacenamiento clave/valor basado en Raft en Kubernetes.

Descripción de la Práctica

La práctica parte de la base de la Práctica 4, donde se implementó el protocolo Raft completo. Ahora se deberá adaptar el código del servidor Raft que ejecuta cada una de las réplicas y también el del cliente Raft, de tal modo que se pueda trabajar con estos a través de Kubernetes.

También se deberá preparar la máquina, instalando `docker`, `kind` y `kubectl` para poder desarrollar la práctica. Para ello viene bien la explicación sobre la puesta en marcha de Kubernetes y los diferentes comandos que puede ser útiles que vienen en los anexos de los enunciados de la práctica. Además, también incluye referencias a sitios de interés.

Por otro lado, se dispone de ejemplos iniciales para entender el funcionamiento de las diferentes herramientas en el fichero comprimido de la práctica.

Desarrollo de la puesta en marcha de las aplicaciones en Kubernetes

Lo primero que se ha realizado ha sido la instalación de `docker`, seguido de `kind` y `kubectl` en la máquina para poder comenzar con las pruebas iniciales.

Una vez está preparado se emplea `./kind-with-registry.sh` para arrancar el cluster completo.

Una vez hecho esto, se han realizado las pruebas pertinentes para entender el funcionamiento de todos los controladores (`pods_go.yaml`, `statefulset.yaml` y `deploy_go.yaml`) con el fin de encontrar el mejor para el servidor raft que queremos poner en marcha.

Podría resumirse de la siguiente manera:

- **Pods básicos**:
  - Son unidades definidas manualmente, su nombre y dirección IP pueden cambiar al recrearlos por lo que no son adecuados para la operativa de los servidores raft.
- **Pods Deployment**:
  - Estos pods sirven para conjuntos de réplicas sin estado en los que todos son idénticos.
  - Además, la identidad no está garantizada al reiniciarse.
  - De la misma manera que antes, no son adecuados para la operativa de los servidores raft.
- **Pods StatefulSet**:
  - Se trata de un recurso que permite la identidad fija por réplica.
  - En el cada nodo tiene un nombre predecible y mantiene su identidad tras reinicios
  - Además, cada pod cuenta con direcciones DNS únicas por pod, por medio de la etiqueta `raft.default.svc.cluster.local` (p.e. `nodo-0.raft.default.svc.cluster.local`).
  - Por último, no necesita definir todos los pods manualmente.

Por todo lo dispuesto anteriormente, queda claro cual es la mejor opción para las **réplicas** raft, `StatefulSet`.

Para el caso del **cliente** raft, basta con definir un **pod básico** ya que, es el usuario quien va a interactuar con él y, por tanto, no requiere ninguna de las ventajas que dan los otros controladores.

Implementación

Para poner en marcha el servicio, se necesita crear un contendor con la imagen del servidor y la del cliente, para ello, se han desarrollado dos scripts, `recompileClient.sh` y `recompileServer.sh` que se encargan de compilar el cliente y servidor, preparar los contenedores y subirlos al registro.

Una vez hecho esto, tenemos que poner en marcha los pods de las réplicas y el cliente. Para ello, se debe ejecutar el script `go_stateful.sh`, que se encarga de poner en marcha lo definido en el fichero `statefulset_go.yaml`. Este último, ha sido modificado para desplegar las réplicas en el puerto 6000 y el cliente en el 7000 como está definido en los Dockerfiles.

Al ejecutar el script ya tenemos el servicio raft puesto en marcha en Kubernetes y podemos realizar así las pruebas que consideremos oportunas.

Pruebas

Se ha comprobado que cada nodo se ha puesto en marcha en un worker distinto:

```
NAME      READY   STATUS    RESTARTS      AGE   IP           NODE           NOMINATED NODE   READINESS
cliente   1/1     Running   0             47m   10.244.1.2   kind-worker3   <none>           <none>
nodo-0    1/1     Running   2 (49s ago)   47m   10.244.2.2   kind-worker4   <none>           <none>
nodo-1    1/1     Running   3 (46s ago)   47m   10.244.4.2   kind-worker2   <none>           <none>
nodo-2    1/1     Running   3 (43s ago)   47m   10.244.3.2   kind-worker    <none>           <none>
```

Se han realizado pruebas para probar el funcionamiento de la elección de lider, deconectando el nodo líder:

```
Comandos disponibles:
  estado <nodo>                     # Estado de un nodo específico
  estadoReplicacion <nodo>          # Estado de Log y Almacen de un nodo
  detener <nodo>                    # Detener un nodo
  leer <lider> <clave>              # Leer valor de una clave
  escribir <lider> <clave> <valor>  # Escribir valor en una clave
  exit                              # Salir del cliente
$ estado 1

[Información del Nodo]
Nodo: 1 | Mandato: 78 | EsLíder: true | Líder: 1

Comandos disponibles:
  estado <nodo>                     # Estado de un nodo específico
  estadoReplicacion <nodo>          # Estado de Log y Almacen de un nodo
  detener <nodo>                    # Detener un nodo
  leer <lider> <clave>              # Leer valor de una clave
  escribir <lider> <clave> <valor>  # Escribir valor en una clave
  exit                              # Salir del cliente
$ estado 2

[Información del Nodo]
Nodo: 2 | Mandato: 78 | EsLíder: false | Líder: 1

Comandos disponibles:
  estado <nodo>                     # Estado de un nodo específico
  estadoReplicacion <nodo>          # Estado de Log y Almacen de un nodo
  detener <nodo>                    # Detener un nodo
  leer <lider> <clave>              # Leer valor de una clave
  escribir <lider> <clave> <valor>  # Escribir valor en una clave
  exit                              # Salir del cliente
$ detener 1
Nodo 1 detenido correctamente.

Comandos disponibles:
  estado <nodo>                     # Estado de un nodo específico
  estadoReplicacion <nodo>          # Estado de Log y Almacen de un nodo
  detener <nodo>                    # Detener un nodo
  leer <lider> <clave>              # Leer valor de una clave
  escribir <lider> <clave> <valor>  # Escribir valor en una clave
  exit                              # Salir del cliente
$ estado 0

[Información del Nodo]
Nodo: 0 | Mandato: 79 | EsLíder: true | Líder: 0
```

También se ha probado la replicación de los nodos:

```
Comandos disponibles:
```

```
  estado <nodo>                      # Estado de un nodo específico
  estadoReplicacion <nodo>           # Estado de Log y Almacen de un nodo
  detener <nodo>                     # Detener un nodo
  leer <lider> <clave>               # Leer valor de una clave
  escribir <lider> <clave> <valor>   # Escribir valor en una clave
  exit                               # Salir del cliente
$ estadoReplicacion 0
Log (len=4):
  [0] {0 0 {start  }}
  [1] {1 3 {escribir 0x0000 chocolate}}
  [2] {2 3 {escribir 0x0000 chocolate}}
  [3] {3 3 {leer 0x0000 }}
------ Almacén (claves limitadas) ------
  0x0000: chocolate
========================================

Comandos disponibles:
  estado <nodo>                      # Estado de un nodo específico
  estadoReplicacion <nodo>           # Estado de Log y Almacen de un nodo
  detener <nodo>                     # Detener un nodo
  leer <lider> <clave>               # Leer valor de una clave
  escribir <lider> <clave> <valor>   # Escribir valor en una clave
  exit                               # Salir del cliente
$ estadoReplicacion 1
Log (len=4):
  [0] {0 0 {start  }}
  [1] {1 3 {escribir 0x0000 chocolate}}
  [2] {2 3 {escribir 0x0000 chocolate}}
  [3] {3 3 {leer 0x0000 }}
------ Almacén (claves limitadas) ------
  0x0000: chocolate
========================================
```

Incluso cuando se desconecta un nodo:

```
Comandos disponibles:
  estado <nodo>                      # Estado de un nodo específico
  estadoReplicacion <nodo>           # Estado de Log y Almacen de un nodo
  detener <nodo>                     # Detener un nodo
  leer <lider> <clave>               # Leer valor de una clave
  escribir <lider> <clave> <valor>   # Escribir valor en una clave
  exit                               # Salir del cliente
$ estado 0

[Información del Nodo]
Nodo: 0 | Mandato: 3 | EsLíder: false | Líder: 1

Comandos disponibles:
  estado <nodo>                      # Estado de un nodo específico
  estadoReplicacion <nodo>           # Estado de Log y Almacen de un nodo
  detener <nodo>                     # Detener un nodo
  leer <lider> <clave>               # Leer valor de una clave
  escribir <lider> <clave> <valor>   # Escribir valor en una clave
  exit                               # Salir del cliente
$ estadoReplicacion 0
Log (len=1):
  [0] {0 0 {start  }}
------ Almacén (claves limitadas) ------
========================================

Comandos disponibles:
  estado <nodo>                      # Estado de un nodo específico
  estadoReplicacion <nodo>           # Estado de Log y Almacen de un nodo
```

```
  detener <nodo>                    # Detener un nodo
  leer <lider> <clave>              # Leer valor de una clave
  escribir <lider> <clave> <valor>  # Escribir valor en una clave
  exit                              # Salir del cliente
$ leer 1 0x0000
----- RESPUESTA REMOTA -----
- ValorADevolver: chocolate
- IndiceRegistro: 4
- Mandato: 3
- EsLider: true
- IdLider: 1
----- FIN RESPUESTA REMOTA -----

Comandos disponibles:
  estado <nodo>                     # Estado de un nodo específico
  estadoReplicacion <nodo>          # Estado de Log y Almacen de un nodo
  detener <nodo>                    # Detener un nodo
  leer <lider> <clave>              # Leer valor de una clave
  escribir <lider> <clave> <valor>  # Escribir valor en una clave
  exit                              # Salir del cliente
$ estadoReplicacion 0
Log (len=5):
  [0] {0 0 {start  }}
  [1] {1 3 {escribir 0x0000 chocolate}}
  [2] {2 3 {escribir 0x0000 chocolate}}
  [3] {3 3 {leer 0x0000 }}
  [4] {4 3 {leer 0x0000 }}
------ Almacén (claves limitadas) ------
  0x0000: chocolate
========================================

Comandos disponibles:
  estado <nodo>                     # Estado de un nodo específico
  estadoReplicacion <nodo>          # Estado de Log y Almacen de un nodo
  detener <nodo>                    # Detener un nodo
  leer <lider> <clave>              # Leer valor de una clave
  escribir <lider> <clave> <valor>  # Escribir valor en una clave
  exit                              # Salir del cliente
$ estadoReplicacion 1
Log (len=5):
  [0] {0 0 {start  }}
  [1] {1 3 {escribir 0x0000 chocolate}}
  [2] {2 3 {escribir 0x0000 chocolate}}
  [3] {3 3 {leer 0x0000 }}
  [4] {4 3 {leer 0x0000 }}
------ Almacén (claves limitadas) ------
  0x0000: chocolate
========================================
```

Conclusiones

Las pruebas realizadas demuestran que la implementación cumple con los requisitos del algoritmo Raft extendido en la Práctica 4: se elige un único líder, se detectan y recuperan fallos de nodos, y las operaciones de lectura y escritura se replican y comprometen correctamente en la mayoría de nodos. El sistema es robusto ante fallos y garantiza la consistencia de los datos replicados y del almacén clave-valor.