Proyecto en Docker-Compose y Kubernetes

Joel M. García Escribano

Diciembre 2022

RethinkDB

Índice

1. RethinkDB

RethinkDB es una base de datos open source que se enfoca en ofrecer una solución de base de datos escalable y fácil de usar para aplicaciones en tiempo real. Algunas de sus características más importantes son:

- Soporte para consultas en tiempo real, lo que permite a los usuarios recibir actualizaciones en tiempo real cuando los datos cambian en la base de datos.
- Un motor de consultas flexible que permite realizar consultas complejas de manera sencilla.
- Integración con varios lenguajes de programación, como JavaScript, Python, Ruby y otros.
- Alta disponibilidad para garantizar la integridad de los datos.

RethinkDB también ofrece una serie de características adicionales, una consola de administración web y una API de lenguaje de consultas que permite a los desarrolladores crear aplicaciones personalizadas que se integren con la base de datos. En general, RethinkDB es una excelente opción para aquellos que buscan una base de datos potente y fácil de usar para sus aplicaciones en tiempo real.

Fue fundada en 2009[1] por la empresa *Y Combinator*[1] y puesta en código abierto en 2012. Al no poder mantener económicamente el proyecto, terminó en 2016 y en 2017 la fundación Cloud Native Computing Foundation adquirió las licencias del código y apoyó su desarrollo

Utiliza el lenguaje de consultas interno NoSQL llamado ReQL y permite realizar funciones, agrupaciones y más funciones comunes de las que disponen lenguajes similares

El sistema de consultas está diseñado para aceptar consultas HTTP en tiempo real, por lo que se recomienda utilizar la base de datos en los siguientes casos:

- Juegos multijugador
- Mercados en línea en tiempo real
- Aplicaciones de streaming de datos
- Aplicaciones móviles
- Aplicaciones web colaborativas

Proporciona una alta escalabilidad de las funciones y una alta disponibilidad de la base de datos con funciones automáticas para asegurar su fiabilidad.

Sin embargo, no se recomienda su uso si se quieren utilizar esquemas que sigan todos los principios ACID. En estos casos se recomienda utilizar gestores de bases de datos relacionales como PostgreSQL o MySQL. Tampoco se recomienda si se quiere crear colecciones que ocupen muy poco, ya que el tamaño mínimo que ocupan las mismas es de 10 MB.

RethinkDB está siendo utilizada actualmente por cientos de startups [2] que la emplean para aplicaciones de diferentes ámbitos. Como es de código abierto, cuenta con multitud de desarrolladores que han contribuido a crear nuevas versiones durante toda su historia.

2. Explicación

Se pide realizar una serie de tareas con las herramientas de creación y orquestración de contenedores[2] Docker y Kubernetes y con la aplicación de gestión de bases de datos orientada a documentos JSON rethinkDB. Se cumplen todos los requisitos pedidos, se proveen todos los archivos utilizados en el desarrollo del trabajo y un script *inicializarTodo* para llevar a cabo de forma automática las instrucciones necesarias de Kubernetes.

[1] Combinator (YC) es una aceleradora de startups estadounidense, fundada en marzo del 2005

[2] Un contenedor crea una capa de virtualización aislada a partir del sistema operativo anfitrión

2.1. Docker-Compose

- Ejecutando *docker-compose up* dentro de la carpeta Archivos Docker se levantan los container de rethinkDB y del script. Primero se levanta la base de datos y después le sigue el script, que crea una colección para luego insertar documentos y leerlos. Cuando ya se ha creado la colección anteriormente, no se vuelven a insertar los documentos y se borra la colección para crearla en la siguiente ejecución. Esto se hace para evitar insertar los mismos documentos cada vez que se levantan los contenedores y así tener siempre 3 documentos en la base de datos.
- La imagen de rethinkDB es la oficial y la imagen del script creada por mi es una imagen de alpine a la que se le instala python, pip y la librería de python de rethinkDB. Esta imagen está alojada en mi repositorio de docker.hub *korah91/rethinkdbscript-docker*

2.2. Kubernetes

- Se ha creado una aplicación Kubernetes que cumple con las funcionalidades de la aplicación en docker-compose. Además de eso, se han añadido más funcionalidades adicionales. El clúster está formado por 3 deployments, 2 servicios de tipo Cluster-IP, una reclamación de volumen y un objeto Ingress.

2.2.1. Deployments

- Son 3 deployments en total; el que contiene la base de datos rethinkDB, el que ejecuta el script y el que levanta un servidor NGINX.
- He decidido utilizar NGINX para poder acceder a la interfaz de la base de datos desde él y para tener un container con el que comprobar que el volumen persistente funciona correctamente.
    - Como se explica en el script en Bash *inicializarTodo.sh*, para comprobar que el volumen está montado en el container de NGINX y en el de rethinkDB, se accede a la shell de NGINX y en la ubicación */usr/share/nginx/html/rethinkdb* está la carpeta montada por el volumen persistente.
- Dicho script también aporta al final la dirección IP en la que está escuchando el servidor NGINX. Al dirigirse a esa IP se puede viajar a la interfaz de la base de datos.
- El deployment script levanta una imagen parecida a la utilizada en docker-compose y se encuentra en el repositorio *korah91/rethinkdbscript*. El pod realiza las consultas a la base de datos comunicándose con el Cluster-IP del deployment de rethinkDB y sigue el siguiente proceso:
    - La primera vez que se levanta un pod en el deployment se conecta a rethinkDB, crea una colección y se termina el script, terminando también el ciclo de vida del pod.
    - Cuando el deployment recibe la señal de cierre del pod, lo pone en el estado *CrashLoopBackOff* para reiniciarlo con más retardo cada vez.
    - Las siguientes veces que se ejecute el script detectará que ya existe una tabla porque se ha guardado en el volumen persistente y procederá a leer los 3 documentos escritos. De esta forma además de la anteriormente mencionada con NGINX se puede comprobar que el volumen persistente funciona.

2.2.2. Volumen Persistente

El volumen persistente se ha creado implementando una reclamación de volumen *PersistentVolumeClaim*[3]. Tiene la posibilidad de poder leerse y escribirse por más de un pod a la vez para el caso en el que se escriba algo en la base de datos que acabe en el volumen y a la vez se modifiquen o lean los contenidos desde el contenedor NGINX (accediendo al montaje del volumen desde su shell).

- Se ha montado en la carpeta de rethinkDB donde se guardan las tablas y en una carpeta de NGINX.

2.2.3. Cluster-IP

Se utilizan 2 Cluster-IP; uno en rethinkDB para que el script pueda modificar la base de datos y para poder ser redirigido desde el ingress y otro en NGINX para ser también redirigido desde el ingress.

- RethinkDB necesita usar el puerto 28015 (driver) para las consultas desde script y la interfaz se sirve en el puerto 8080 (interfaz)[4]. Se redirige desde el puerto 80 al 8080 para que el objeto ingress lo utilice.
- NGINX solo utiliza el puerto 80 y es para el ingress.

2.2.4. Ingress

Finalmente, el objeto ingress se utiliza para acceder al servidor NGINX y a la interfaz de la base de datos desde un navegador.

- Como este proyecto se ha realizado utilizando el clúster de Minikube, han sido necesarias algunas modificaciones al ingress en el campo *annotations* y en las rutas.
- Se definen dos rutas; una vacía para acceder al NGINX y otra llamada rethinkdb para acceder a la interfaz de rethinkDB.
- Para que la primera página de NGINX mostrara las rutas, se ha utilizado una imagen de NGINX modificada por mí alojada en el repositorio *korah91/nginx* y que se construye a partir del fichero *Dockerfile.nginx* que habilita el *autoindex* de NGINX[5] y borra la página que se mostraría por defecto.

2.2.5. Namespaces

Como herramienta de Kubernetes no dada en clase he implementado el uso de *Namespaces*.

- Un *Namespace* es una división lógica del clúster de Kubernetes que también se puede definir como un clúster virtual[6]. Se suelen utilizar para no saturar un clúster con objetos de Kubernetes que puedan tener nombres iguales o parecidos y que tengan configuraciones diferentes.
- Por ejemplo, en el caso de tener dos *Namespaces*, uno de desarrollo y otro de producción, los deployments se llamarán igual y cuando se levante un deployment en un Namespace no se sobreescribirá sobre el del otro. Si solo hubiese un *Namespace* y hubiese un deployment llamado myapp-deployment que fuese la versión de producción, cuando los desarrolladores quisieran probar una versión en desarrollo de la misma aplicación tendrían que sustituir el deployment de producción. Esto es claramente un error grave que no debería ocurrir y justifica el uso de *Namespaces*.
- Al ejecutar el script *inicializarTodo.sh* se pide escribir un nombre para el *Namespace* que se va a utilizar. Todos los objetos de Kubernetes serán creados dentro de ese clúster virtual y si se ejecutase otra vez poniendo un nombre diferente no deberían haber conflictos.
- Sin embargo, hay componentes como los volúmenes que no pueden pertenecer a un *Namespace* y pertenecen al clúster global. De todas formas no ha supuesto un problema en este proyecto.
- Mi implementación de los *Namespaces* es muy sencilla, pero podría desarrollarse mucho más en un caso real en el que se necesite autorización para crear elementos en un *Namespace* o se necesite conectar objetos de diferentes clústeres virtuales

2.2.6. Diagrama Kubernetes

Se dispone de un diagrama para una mejor visualización de los objetos utilizados en Kubernetes y el clúster generado. Ver figura 1

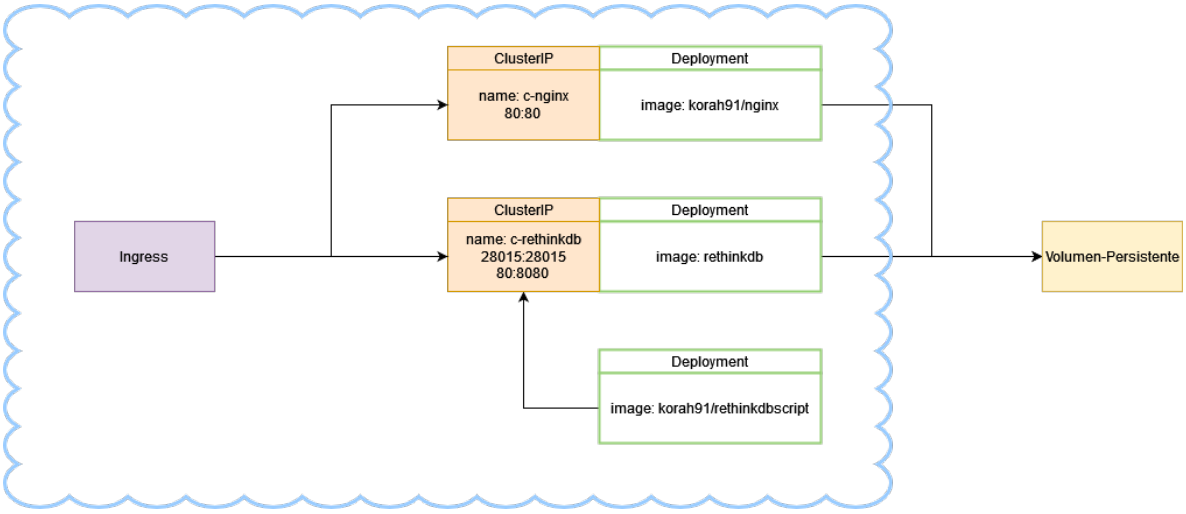

Figura 1: Diagrama del clúster de Kubernetes.

3. Enlaces

- Imágenes:
  - script en Docker: https://hub.docker.com/r/korah91/rethinkdbscript-docker
  - script en Kubernetes: https://hub.docker.com/r/korah91/rethinkdbscript
  - nginx en Kubernetes: https://hub.docker.com/r/korah91/nginx
- Repositorio GitHub:
  - Github: https://github.com/korah91/DockerRethinkDB

Referencias

[1] YC-Funded RethinkDB, A MySQL Storage Engine Built From The Ground Up For Solid State Drives. Recuperado 1 de diciembre de 2022, de https://techcrunch.com/2009/07/28/yc-funded-rethinkdb-a-mysql-storage-engine-built-from-the-ground-up-for-ssds/.

[2] Frequently asked questions. Recuperado 27 de noviembre de 2022, de https://rethinkdb.com/faq

[3] Kubernetes Volumes explained. Recuperado 29 de noviembre de 2022, de https://www.youtube.com/watch?v=0swOh5C3OVM

[4] Repositorio de la imagen de rethinkDB para Docker. Recuperado 22 de noviembre de 2022, de https://hub.docker.com/_/rethinkdb/

[5] Repositorio de NGINX modificado para habilitar autoindex. Recuperado 2 de diciembre de 2022, de https://github.com/FreedomBen/nginx-docker

[6] Namespaces. Kubernetes. Recuperado 1 de diciembre de 2022, de https://kubernetes.io/docs/concepts/overview/working-with-objects/namespaces/