ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS

IMPLEMENTACIÓN DE CONTENEDORES ALOJADOS EN LA NUBE

IMPLEMENTACIÓN DE UNA BASE DE DATOS MEDIANTE CONTENEDORES ALOJADOS EN LA NUBE

TRABAJO DE INTEGRACIÓN CURRICULAR PRESENTADO COMO REQUISITO PARA LA OBTENCIÓN DEL TÍTULO DE TECNÓLOGO SUPERIOR EN REDES Y TELECOMUNICACIONES

WILSON DAVID TORRES CAÑAR

DIRECTOR: ING. GABRIELA KATHERINE CEVALLOS SALAZAR, MSC.

DMQ, agosto 2022

CERTIFICACIONES

Yo, Wilson David Torres Cañar declaro que el trabajo de integración curricular aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

Wilson David Torres Cañar

wilson.torres@epn.edu.ec

wilsondavtorres@gmail.com

Certifico que el presente trabajo de integración curricular fue desarrollado por Wilson David Torres Cañar, bajo mi supervisión.

Gabriela Katherine Cevallos Salazar

DIRECTOR

gabriela.cevalloss@epn.edu.ec

DECLARACIÓN DE AUTORÍA

A través de la presente declaración, afirmamos que el trabajo de integración curricular aquí descrito, así como el (los) producto(s) resultante(s) del mismo, son públicos y estarán a disposición de la comunidad a través del repositorio institucional de la Escuela Politécnica Nacional; sin embargo, la titularidad de los derechos patrimoniales nos corresponde a los autores que hemos contribuido en el desarrollo del presente trabajo; observando para el efecto las disposiciones establecidas por el órgano competente en propiedad intelectual, la normativa interna y demás normas.

WILSON DAVID TORRES CAÑAR

DEDICATORIA

Este proyecto está dedicado en toda su extensión a mis padres los cuales fueron mi base e inspiración para poder realizar este objetivo en mi vida. Ellos fueron, son y serán siempre un gran sustento para mí de una manera profunda e importante, ya que con sus enseñanzas, conocimientos y experiencias siempre van a ser los cimientos de mi crecimiento académico y personal.

Este arduo trabajo asimismo va dedicado a mis hermanos que siempre me ayudaron y apoyaron para poder salir adelante pese a todo, y a poder ser realista para afrontar todos los obstáculos que pone la vida y el destino.

Wilson David Torres Cañar

AGRADECIMIENTO

Es muy importante agradecer de manera cordial a mis padres los cuales fueron mi motor y mi impulso para poder seguir adelante. Siempre estaré agradecido por su consejo y su sustento incondicional a mi familia y a mí. También al apoyo emocional frente a cualquier impedimento o bache que pueda presentarse en el camino de mi vida personal y académica.

Así también de manera afable agradezco mucho a determinados maestros/as que han confiado en mí, y realmente me han apoyado con sus conocimientos y sabiduría. A la vez de poder así demostrarlo dentro y fuera de la materia, y poder apreciar claramente su efusión por instruir.

Agradezco asimismo de manera muy atenta a mi directora por saber tutelarme, recomendarme y apoyarme en este objetivo en mi vida. Ya que fue de mucha ayuda a tener su pauta, para así poder concluir este proyecto de manera idónea.

Wilson David Torres Cañar

ÍNDICE DE CONTENIDOS

RESUMEN

El uso de los contenedores en la actualidad ha hecho que el despliegue, control, organización, almacenamiento y la optimización de diferentes servicios mediante aplicaciones sea más sencilla y eficaz; en función de los requerimientos de una persona o empresas. Los contenedores conjuntamente con el uso de la nube, da innumerables beneficios que los proveedores brindan para ejecutar, almacenar y controlar varios servicios.

Este proyecto presenta la utilidad de las bases de datos mediante contenedores dentro de la nube, donde se puede trabajar y recopilar los datos instantáneamente sin requerir demasiados recursos físicos. En su primer apartado, se detalla la explicación del componente y la especificación de los objetivos, que son la base para el desarrollo del proyecto de titulación.

El segundo apartado describe la metodología donde se toma en cuenta el procedimiento de realización del proyecto, con el fin de cumplir cada objetivo. Llegando a ejecutar y albergar el contenedor de base de datos en la nube, con las opciones que brindan los proveedores de nube.

En el tercer apartado se constata el avance de cada objetivo, así como las pruebas de verificación y los resultados que se lograron obtener.

Asimismo, en el apartado de conclusiones se tomó lo experimentado en el desarrollo de los objetivos, y a su vez en recomendaciones se detallan los desafíos que se presentaron y se solventaron a lo largo del proyecto.

Concluyentemente en el apartado de bibliografía se presentan las fuentes bibliográficas en las cuales se basó el proyecto.

PALABRAS CLAVE: Contenedor, base de datos, proveedores de nube, RethinkDB, Docker.

ABSTRACT

The use of containers today has made it easier and more efficient to deploy, control, organize, store and optimize different services through applications; depending on the requirements of a person or companies. Containers, along with the use of the cloud, provide myriad benefits that providers provide to run, store, and control various services.

This project introduces the utility of containerized databases within the cloud, where you can work and collect data instantly without requiring too many physical resources. The first section details the explanation of the component and the specification of the objectives, which are the basis for the development of the titling project.

The second section describes the methodology where the procedure for carrying out the project is taken into account, in order to meet each objective. Get to run and host your database container in the cloud, with options provided by cloud providers.

The third section shows the progress of each objective, as well as the verification tests and the results achieved.

Likewise, in the conclusions section, what was experienced in the development of the objectives was taken, and in turn, the recommendations detail the challenges that arose and were solved throughout the project.

Conclusively, the bibliographical sources on which the project was based are presented in the bibliography section.

KEY WORDS: *Container, database, cloud providers, technology, RethinkDB, Docker.*

1 DESCRIPCIÓN DEL COMPONENTE DESARROLLADO

El presente proyecto conlleva una forma diferente de contextualizar las redes de información y la implementación de servicios, procesos y aplicativos; con el fin de realizar una automatización de microservicios, procesos de *software*, servidores y aplicaciones se ejecutará el despliegue de una base de datos mediante un contenedor alojado en la nube.

Para implementar los contenedores se procederá a analizar algunas herramientas de DevOps con el fin de seleccionar la más adecuada para ejecutar el presente proyecto. Además, para disminuir los costos en cuanto al *hardware* requerido y con el fin de modernizar la implementación de contenedores se pretende alojar en la nube a los mismos, donde se analizará entre las opciones de nube sus características, ventajas y desventajas seleccionando el proveedor apropiado para este fin.

Para mejor explotación de todas las herramientas y ayudas dentro de la nube, se optó por alojar ahí a los contenedores con los beneficios propios que la plataforma ofrece, siendo parte esencial para el desenvolvimiento correcto del contenedor.

1.1 Objetivo general

Implementar diferentes servicios mediante contenedores alojados en la nube.

1.2 Objetivos específicos

- Analizar algunas herramientas de DevOps para realizar contenedores.
- Analizar algunos proveedores de nube para alojar contenedores.
- Implementar los contenedores y alojarlos en la nube seleccionada.
- Realizar pruebas de funcionamiento y verificación de los resultados obtenidos.

1.3 Alcance

El presente proyecto consiste en analizar algunas de las herramientas DevOps que permitan realizar contenedores, se seleccionará a la más adecuada con el fin de implementar un contenedor de una base de datos. Además, se procederá a seleccionar de diferentes proveedores de nube al más idóneo para alojar a los contenedores. Se realizarán pruebas de funcionamiento y verificación de los resultados obtenidos.

1.4 Marco Teórico

Contenedores

Se los puede definir como una tecnología establecida que se centra en el encapsulamiento de aplicaciones, procesos, entornos informáticos y/o servicios determinados; estos empaquetan todas las herramientas para su funcionamiento y ejecución como código fuente, variables importantes de entorno, librerías, archivos de configuración y demás dependencias [1] [2].

Esta clase de técnica deriva de la virtualización de un sistema operativo (SO), la cual funciona como un paquete o unidad ejecutable en cualquier entorno, de un *software* [3]. Donde para que su funcionamiento sea transparente del resto del sistema se aíslan todos sus procesos, los cuales también se optimizan para que su acceso a la cantidad de memoria, disco y CPU sea menor [3] [4].

La finalidad de los contenedores radica en su utilización dentro de cualquier ambiente que puede ser desde un escritorio o computadora independiente de su SO, o dentro de las disposiciones de las Tecnologías de la Información (TI) centrado en el manejo de aplicaciones y procesos de automatización, hasta trabajos en la Nube [3]. Esto contribuye al desarrollo de los profesionales dentro del mundo de la informática ya que brinda la facilidad de ejecutar cualquier "cosa" [1].

Dentro de sus ventajas adicionales están que son ligeros debido a que eliminan procesos adicionales dentro de un SO por cada aplicación, lo cual se deriva de compartir el *kernel* de la máquina [3]. También son portables y autónomos de la plataforma, donde los contenedores tienen la capacidad de adaptarse a cambios o arquitecturas modernas [3]. Además, son más eficaces en cuestión de utilización de recursos de la computadora de manera consistente y escalable a diferencia de una aplicación o SO desde cero [3]. Asimismo, también se destaca su agilidad porque son constantemente actualizados y compilados lo cual permite una mayor evolución de los contenedores [2].

Máquinas Virtuales

Las máquinas virtuales (VM) como tal se implementan mediante un hipervisor, cuya función primordial es separar los recursos de la máquina huésped o física para alojar a una VM con un diferente SO [4]. Los hipervisores tienen beneficios como la implementación de varios SO en un mismo entorno físico, y el aprovechamiento de servidores con más rapidez y el uso adecuado de recursos [2].

La implementación de máquinas virtuales tiene ventajas, de las cuales se pueden destacar que son fáciles de manejar, relativamente más asequibles, de gestión eficaz y cómoda instalación. Además, dan más seguridad al sistema físico, porque si algún *malware* invade la VM y logra dañarla, no afecta a la máquina en la que reside [5].

Sin embargo, las falencias de las máquinas virtuales radican en su tamaño, ya que pueden llegar a ser muy grandes. En estas máquinas vienen empaquetadas todas sus dependencias como la imagen del SO, código binario, variables, bibliotecas, entre otros [1] [2]. Esto causa una sobrecarga, desborde y/o consumo de recursos de la máquina física, la cual puede colapsar al mantener muchas VM encendidas [1]. Además, el hipervisor provee recursos adicionales como memoria caché para poder guardar los cambios que la preceden, lo que las hace más lentas [4]. Es importante destacar que deben ser actualizadas, monitoreadas y mantenidas de manera constante para que funcionen correctamente [5].

Para enmarcar la comparación de directamente de los hipervisores de manera general, y con un contenedor se muestra la Tabla 1.1 para conocer el contraste correcto que puede tener una máquina virtual con un contenedor.

Tabla 1.1 Hipervisor vs Contenedores [5] [6]

Hipervisor	**Contenedores**
Existe una mayor probabilidad de ocurrencia de algún error debido a su procesamiento y dependencias físicas externas e internas.	Existe una menor probabilidad de ocurrencia de algún error debido a que todo lo necesario para su arranque está en el mismo contenedor.
Un SO se implementa de manera independiente del *hardware* de la máquina física.	Una aplicación se implementa de manera independiente del *hardware* y SO de la máquina física.
Para la creación de VM se necesita gran cantidad de memoria, así como para su instalación y funcionamiento (orden de los Gigabytes).	Para la creación de contenedores se necesita menos cantidad de memoria, así como para su funcionamiento (orden de los Megabytes).
Comparten recursos de la computadora física como memoria, almacenamiento y control de procesos de la misma.	Se pueden ejecutar en cualquier SO mediante un motor para contenedores que les permita correr sobre el mismo.

Continúa

Hipervisor	Contenedores
Puede arrancar varios SO de tipo bare-metal (parte superior de un servidor), en la parte superior de un SO estándar o de manera alojada o aislada.	Tienen entre sus facilidades la portabilidad absoluta debido a que todas sus dependencias están en el mismo contenedor.
Posee una mayor seguridad debido a que es por completo un SO.	Posee una menor seguridad debido a su ligereza, aunque se remedia con su implementación dentro de la Nube.

Tras el análisis realizado de las máquinas virtuales en comparación con los contenedores, es importante recalcar que la utilización de los contenedores va a ser de vital importancia, debido a su rapidez, eficacia y ligereza que puede desarrollar la red para dar un servicio.

Se rescata que las máquinas virtuales y los contenedores son de gran ayuda en función del ambiente o las necesidades que la actividad lo requiera; donde los contenedores por su peso ligero y su mínima probabilidad de presentar errores tiene compatibilidad con el entorno en el cual se lo implemente, hace que se vuelva funcional, eficaz, adaptable sin falencias e independiente del sistema operativo.

En cuanto al uso de la nube es importante recalcar que utilizar contenedores es más apropiado y fácil de realizar, ya que proporciona cierta autonomía de sus servicios en cuestión de *hardware* y da un balanceo de cargas de trabajo en cuestión de *software*. Además, el proveedor de la nube da las funcionalidades, así como modularidad en los procesos para poder cumplir la ejecución y almacenamiento de los contenedores [4] [7].

Sistemas de Gestión de Bases de Datos

También llamado gestor de base de datos o sistema de gestión de base de datos (SGBD), se la define como la pauta de una aplicación informática la cual tiene la capacidad de manipular y operar una base de datos (BD). Esta aplicación por lo tanto tiene una interfaz para que el usuario y/o administrador pueda usar la información de la BD, con la constancia del manejo de los niveles de análisis, consulta, modificación y almacenamiento. Dando con estos cuatro niveles la facultad de granularidad (constancia) de los datos, centrándose en la operación y manipulación eficaz de los mismos [8].

Cabe mencionar que para que el SGBD pueda implementarse y utilizarse de manera adecuada debe poseer un hilo conductor que pueda dar la capacidad de relación, organización y estructura a los datos, con el fin de crear un entorno de factible ingreso y manipulación de la BD. Destacando que el hilo conductor da autonomía de la plataforma informática en el cual trabaja el SGBD [8].

Asimismo, el gestor de base de datos será el que proporcione ayudas o instrumentos para el control de la misma, desde la perspectiva del cliente o usuario. Estas herramientas tales como sistemas de búsqueda y/o generador de informes, deben asegurar que la información recopilada tenga siempre seguridad, integridad y consistencia sin ninguna clase de manipulación [8].

La clasificación del SGBD radica en la dependencia de parámetros del modelo de datos, en la Tabla 1.2 se observa un breve repaso de algunas clases de SGBD que hay en función de su modelo y características.

Tabla 1.2 Clasificación de SGBD [8]

SGBD	Características
Relacional	Este en la actualidad es el más común y popular, donde se intuye que la información, dentro del mismo, mantiene cierta relación bidireccional (columnas y filas).
Multidimensionales	Similar al relacional con la singularidad de tener una cantidad irresoluta de filas y columnas, para así mostrar un infinito número de relaciones y dimensiones.
Deductivas	Este modelo parte de instancias, junto con la aplicación de reglas y medios para dar deducciones a la información de la BD.
Jerárquico	Su modelo es en tipo árbol, puede almacenar muchos datos, donde no tiene flexibilidad, aunque haya redundancia en la información.
Orientada a Objetos	Modelo más actual, donde su objetivo radica en almacenar información junto con el concepto de paradigma a objetos complejos (imágenes, voz, texto, gráficas, entre otros).
Documentales	Su modelo sirve para indexar de manera completa texto, con la finalidad de manejar grandes cantidades de información.
Transaccionales	Su modelo se enfoca principalmente en el traslado de información de envío y recepción de manera transparente, eficiente y rápida.

Una vez visto las diferentes variantes de bases de datos, se aclara que los SGBD pueden estar en diferentes aplicaciones las cuales son adaptativas, a las diferentes circunstancias y necesidades del usuario en cada actividad que realice. Estos programas se pueden clasificar en dos tipos: los que son relacionales o SQL (MySQL, Microsoft SQL Server, MariaDB, SQLite, entre otros) y los que son no relacionales o NoSQL (RethinkDB, MongoDB, Redis, Casandra, entre otros) [8].

- **MySQL**

Este es un SGBD relacional, tiene las características que posee múltiples hilos y puede trabajar con varios usuarios (multi usuario). Este es uno de los más populares para las páginas *web* en la actualidad, además siendo el más solicitado para desarrollar aplicaciones de *software* libre. Está licenciado bajo *General Public License* (GNU GPL), que es de código abierto y *software* libre, por lo que da la potestad de manera libre de estudiarla, utilizarla, alterarla y compartirla [9].

Entre sus beneficios es que es de uso factible y posee un gran rendimiento, además es sencillo de implementar y configurar en una máquina. Sus soportes abarcan desde multiplataforma hasta SSL (*Secure Sockets Layer*) [9].

Sin embargo, una de sus desventajas radica en la adaptabilidad debido a que, frente a una base de datos muy extensa empieza a fallar [9].

- **MariaDB**

Este es un SGBD variante de MySQL, heredando sus características primordiales como ser un sistema multi hilos, funcional con varios usuarios, eficaz, sencillo y de alto rendimiento [8] [9]. Este gestor de bases de datos acoge la ideología de *Open Source* (OSS), donde solo el código es abierto lo cual implica que el mismo puede ser observado, alterado y distribuido de manera provechosa por cualquier persona [9].

Entre sus beneficios se resalta que es totalmente compatible con versiones y extensiones de MySQL, con una mejora en cuanto al mayor almacenamiento de sus motores; posee escalabilidad, mayor seguridad y rapidez en los diferentes procesos, y también tiene algunos avances con adaptaciones a BD no relacionales [9].

Sin embargo, una desventaja es la falencia que se puede generar en el cambio de MariaDB a MySQL; además que MariaDB no actualiza regularmente a sus versiones más seguras [9].

- **RethinkDB**

Este es un SGBD no relacional o NoSQL relativamente nuevo, que está orientado a manipular documentos JSON (campos con objetos nombre/valor en pares) planteada para trabajar en la *web* en *Real Time* [10]. Fue lanzado con una licencia Apache 2.0 desarrollada por Apache *Software Foundation* (AFS), la cual acoge la ideología de OSS, donde solo el código es abierto lo cual implica que el mismo puede ser observado, alterado y distribuido de manera provechosa por cualquier persona [11].

Entre sus beneficios están que tiene sencilla implementación, tiene la capacidad del *Changefeeds* (ver los cambios en tiempo real desde cualquier ordenador), tiene un robusto lenguaje de consulta, manejo de promociones automáticamente de clase maestra y tiene la capacidad de la escalabilidad [12].

Entre sus desventajas radica en que no posee ni crea cuentas de usuario y autenticación de los mismos por lo que se tendrá que usar productos de terceros [12].

Al revisar y analizar los programas que son SGBD y/o manejan BD, se elige el gestor de base de datos RethinkDB, debido a que está orientado a la *web. A*demás, que tiene un manejo de datos en tiempo real lo cual lo vuelve más adaptable y eficaz en la manipulación y actualización de los datos almacenados dentro de la BD. Asimismo, tiene una plantilla oficial dentro de la plataforma de *Docker Hub,* esto permite que sea más factible la implementación de este gestor de base de datos como contenedor.

2 METODOLOGÍA

Dentro de este documento se detalla la actividad experimental realizada, partiendo del estudio profundizado del uso y la importancia de los contenedores para el desarrollo de las redes, y la ejecución de aplicaciones. Del mismo modo que se examinó las diferentes herramientas de DevOps para ejecutar el contenedor de base de datos, y también sobre los proveedores en la nube los cuales ayudarán a implementarlos como aplicación.

Con el fin de cumplir con el objetivo general, en primera instancia se analizaron diferentes herramientas DevOps que permiten trabajar con contenedores, obteniendo como conclusión final que la más apta para administrar los contenedores es *Docker*. Esta herramienta posee mucha información y datos relevantes los cuales ayudan a crear, manipular y solucionar problemas en torno a los contenedores.

Luego de la elección de la herramienta para la implementación del contenedor, se realizó un análisis de diferentes plataformas que alojan contenedores en la nube, para

seleccionar la más idónea. En función de la búsqueda se determinó las siguientes plataformas que dan servicios en la nube: Amazon *Web Services* (AWS), Microsoft Azure y Google *Cloud Plataform* (GCP). Además, se analizó su periodo de prueba gratuito, servicios y/o productos útiles, costo del servicio y/o saldo disponible, interfaz visual y facilidad de manejo de cada una. Examinando las opciones y servicios que se pueden tener individualmente, se determinó que la plataforma de GCP es la más apta para alojar este servicio.

En cuanto a la implementación de contenedores se realizó el contenedor de Base de Datos o SGBD RethinkDB, utilizando las herramientas de GCP, la cual proporciona el *Cloud Run* para manipular contenedores. Finalmente se realizaron pruebas de funcionamiento verificando que el contenedor de Base de Datos o SGBD se encuentre alojado en la nube funcionando correctamente.

3 RESULTADOS

Este documento tiene como finalidad implementar una base de datos mediante contenedores y así poder albergarla en una de las plataformas de nube que proveen este servicio. Para poder llevar esto acabó, se utilizó la herramienta de DevOps para el manejo de contenedores *Docker*; y la plataforma de GCP la cual da servicio en la nube. Con el apoyo de las electas se obtuvo como resultado un SGBD RethinkDB subido en la nube, el cual permite guardar y consultar información en grandes cantidades.

3.1 Herramientas de DevOps para realizar contenedores

En la actualidad se puede encontrar un sin número de aplicaciones que pueden ayudar para la creación, implementación y control de los contenedores. Ya que es una tecnología que viene siendo desarrollada desde inicios de los 80, por lo que se analizarán las más populares herramientas para la manipulación de los contenedores los cuales son *Docker*, Podman y Kubernetes.

Docker

Es la tecnología más popular para contenedores, que fue presentada en 2013, cuyos inicios radican en *Docker Engine*. Su principal característica es que tiene integrada la funcionalidad de virtualización, por el núcleo de Linux como *cgroups* (facilita la limitación y aislamiento de recursos) y el *kernel* de Linux de *namespaces* (independización de procesos entre grupos). Esta tecnología tiene como principio retraer las dependencias del *software* y ser autónomo del *hardware*. Donde puede constatar que se puede

virtualizar a nivel de *software* los contenedores, lo que hace que el *software* ejecutable en un contenedor sea adaptable al ambiente en que este se encuentre [13] [6].

Es una herramienta que acoge la ideología de OSS y es libre. Entre sus grandes ventajas están que los contenedores instalados se aíslan del entorno del sistema, constatando y realizando sus procesos de manera transparente. Además, su manejo e implementación es sencilla y es portable, por lo que se puede ejecutar en cualquier dispositivo [13]. Sus contenedores son relativamente seguros debido a su capacidad de aislamiento. Entre sus desventajas, Docker necesita requisitos de *hardware* y *software* específicos para poder funcionar como aplicación en un dispositivo. No soporta ciertas arquitecturas inferiores a 64 bits [14].

Podman

Es una tecnología desarrollada por la corporación *Red Hat*, manejando la filosofía de *Open Source* (código abierto), la cual es un gestor de contenedores y *Pods*, y brinda un ambiente para el control de estos con facilidades para su implementación. Entre sus características es que permite albergar, crear y manipular imágenes de manera eficaz mediante repositorios (almacenes). También permite buscar paquetes e imágenes, los cuales contengan una aplicación específica, donde se pueden descargar en cualquier dispositivo o servidor donde se requiera hacer una determinada actividad [14] [15].

Entre sus beneficios se encuentra el manejo de imágenes de formatos *Open Container Initiative* (OCI) y de *Docker* la cual puede funcionar de manera similar, debido a que su sintaxis es aproximadamente equivalente. No requieren de un *daemon* (demonio) para poder ejecutar un contenedor, por lo que todos los recursos necesarios para gestionar controlar los mismos se realizan de manera descentralizada. Los contenedores se pueden ejecutar sin ser usuario *root* o sin derechos de administrador. Entre sus desventajas están que solo se pueden ejecutar dentro de descendientes de Linux, a la vez que no tiene una API (*Application Programming Interfaces*) de servidor, por lo que solo va utilizar línea de comandos para realizar todos sus procesos. No trabaja con procesos de segundo plano de manera predeterminada ni continuamente [14] [15] [16].

Kubernetes

Igualmente es una tecnología de contenedores de código abierto que fue comercializada en 2014, como resultado de la implementación y amparo de la empresa Google. Es una tecnología versátil la cual puede ofrecer varios tipos de servicios entre estos estás la implementación de microservicios, contenedores o trabajar simulando una nube portátil [13].

Es la más usada para la orquestación de contenedores, lo que significa que puede hacer funcionar a varios contenedores con dependencias y comparticiones de elementos entre ellos, al mismo tiempo que tengan sincronía en sus procesos para cumplir un determinado proyecto. Esta plataforma tiene una excelente documentación y soporte en muchos dialectos incluido español [13]. Sus falencias radican en el traslado de versiones lo cual es complejo; su implementación y clúster es complicado y difícil de manejar. No es compatible con *Docker* en ciertas características (*runtime* directo y soporte) [17].

Una vez analizado y asimilado todas las características, así como pros y contras de cada una de estas diferentes aplicaciones para el manejo de contenedores, se toma en consideración sobre todo la información y soporte de documentación de cada una de estas plataformas. Además de la eficacia de ejecución y rapidez para manipular los contenedores. Entonces se eligió a *Docker* como la mejor opción, debido a que es uno de los más robustos y populares en la actualidad, además puede manipular contenedores. Asimismo, cuenta con una vasta información, imágenes de contenedores oficiales y comunidad que respaldan frente a cualquier dificultad, esto lo vuelve eficaz en muchos aspectos.

3.2 Proveedores de la nube para alojar contenedores

La computación en la nube (*cloud computing*), se la esquematiza como una tecnología que da la facultad de poder entrar a cualquier parte y en cualquier instante de forma remota a diferentes plataformas, aplicaciones, servicios de almacenamiento de registros, ordenadores de información, entre otros sin la necesidad de enlazarse a un computador o servidor [18] [19]. Sus únicos recursos radican en poseer un *web browser* (navegador *web* en español) y una conexión estable a Internet para poder usarlos [18].

Entre sus más grandes ventajas se encuentra la minimización de gastos en cuanto a las plataformas, ahorrar el espacio físico en el cual se encuentra, tendencia a centrar los datos con cierta personalización en base a lo que el usuario final requiere, y el uso de contenedores [19]. A continuación, se va a presentar las diferentes plataformas que trabajan con esta clase de servicio, junto con sus más prominentes características.

Google *Cloud Plataform* (GCP)

Es una plataforma informática desarrollada por la empresa Google, y es un proveedor de recursos en *cloud computing* que se usa para crear, instalar y manipular servicios dentro de Internet. Esta plataforma es eficiente y amplia, cuenta con más de 100 productos y a la vez funciones las cuales ofrecen oportunidades y procedimientos robustos y complejos. Dentro de esta plataforma se ofrecen diferentes servicios lo que

la vuelve apta para soluciones empresariales y/o comerciales, las cuales brindan la capacidad de crear y administrar sitios *web*, controlar procesos, dar almacenamiento de datos, entre otras facultades [20].

Cuenta con niveles de seguridad avanzados, junto con la encriptación que ayuda a evitar la invasión a cualquier sistema y otros ataques como hurto de datos. Entre su gama de productos se encuentran Google *Cloud Storage* que sirve para el almacenamiento de datos, alojamiento de páginas *web* en esta plataforma, Google *Compute Engine* para máquinas virtuales y sus instancias, Google *App Engine* para el desarrollo de aplicaciones, Google *BigQuery* para el almacenamiento de datos en múltiples nubes, Inteligencia artificial y *Machine Learning*, Google *Maps Plataforms*, etc [20].

Hablando de herramientas necesarias existe el *Container Registry* cuya función radica en manipular y salvaguardar de manera idónea imágenes de contenedores de *Docker*. A la vez que cuenta con *Cloud Run* el cual puede procesar e implementar aplicaciones en contenedores con escalabilidad y total control de la misma sin la necesidad de más instancias [21].

Profundizando entre sus ventajas se puede encontrar que tiene un período gratuito bastante amplio de 90 días con un saldo de 300 dólares americanos. Es amigable con el usuario, sus precios son relativamente accesibles, es una infraestructura dedicada a poder brindar desarrollo y escalabilidad, tiene la facultad de hacer una colaboración entre usuarios y proyectos, guarda copias de seguridad de manera segura y protege los datos, entre otras ventajas [5] [20].

Amazon *Web Services* (AWS)

Igualmente es una plataforma que se dedica a proveer servicios en la nube mediante el *cloud computing*, que a la vez permite el acopio de recursos computacionales, aplicaciones, bases de datos, entre otros. Una sus peculiaridades es que, si se requiere de sus servicios completos en recursos y capacidad de infraestructura, se necesita hacer una inversión. Lo que significa que se deberá pagar una suscripción mensual para conectarse a cualquier servicio que ofrece este proveedor en la nube [22]. AWS cuenta con el servicio de Amazon *Elastic Container Registry*, que guarda imágenes de contenedores *Docker*, y ofrece 500 (MB) por mes con 12 meses gratis para las imágenes almacenadas [23].

Entre sus ventajas están su seguridad es que puede procesar los datos en una plataforma externa, que llega a ser beneficiosa en sentido de gastos y cuenta con certificaciones internacionales como la ISO 27001. En torno de las BD que son de gran

importancia, se tiene completa manipulación de las mismas, con la confianza que los datos puedan ser almacenados y operados de manera correcta. Otra ventaja es que los clientes realizarán una baja inversión debido a que todo se conecta a la nube, y se estima que los negocios no deberán realizar un gasto en cuestión de infraestructura física. También se tiene la accesibilidad sencilla y que los servicios pueden absorber la carga de trabajo en cualquier volumen. Otra ventaja es la resiliencia donde cualquier fallo o impedimento que ocurra, la plataforma garantiza un máximo de 4 segundos de estar fuera de servicio [22].

Profundizando en su periodo de prueba y en base a ciertos parámetros, su capa gratuita define como 12 meses gratis (sin saldo) donde se puede acceder desde crear un sitio *web*, compilar datos y ejecutar, desarrollar y probar aplicaciones. Una de sus desventajas radica en que no hay suficiente información o documentación respecto a sus procesos o manera de implementar los servicios de un contenedor, y no es amigable para el usuario con la interfaz [22] [5].

Microsoft Azure

Es una infraestructura privada la cual ofrece servicios en la nube (*cloud computing)*, para los diferentes tipos de administradores y desarrolladores los cuales puedan crear, optimizar, implementar aplicaciones y/u otras herramientas para la comunidad en general. Está plataforma funciona en base a una clasificación de los SO, a la vez que une herramientas de programación y otras tecnologías las cuales son necesarias para los profesionales y creadores de TI [24].

Dentro de las funcionalidades de Microsoft Azure existen las máquinas virtuales, almacenamiento, recuperación y creación de copias de seguridad, procesos de intensivos, App *Service Web*, movilidad multimedia, BD relacionales (SQL) y no relacionales (NoSQL), administración de identidad, *Active Directory*, etc [24].

Entre sus ventajas radican el desarrollo de operaciones en base a un entorno de colaboración, también está que provee una seguridad en la nube con la protección e implementación de los datos e información personal. Ofrece también servicios de infraestructura o plataforma de servicios e incorpora encriptado de todos sus datos. Cuenta con la habilidad de la resiliencia mediante el uso de dispositivos optimizados y eficaces, lo cual baja el gasto, a la vez que permite mejor movilidad y eficacia del manejo de los datos y ofrece también opciones de Inteligencia Artificial [24].

Dentro de sus falencias se encuentran sus altos costos por servicio y su funcionamiento es relativamente lento. Su prueba gratuita solo da 30 días con un saldo acotado en

medida del costo del servicio, el cual no va a ser una gran opción debido a la cantidad de pruebas y tiempo requerido para la implementación del presente proyecto [5].

Una vez analizado y visualizado todos los recursos y características, así como ventajas y desventajas de cada una de estas plataformas, se toma en consideración sobre todo la facilidad de funcionamiento y rapidez en el cual puede implementarse los contenedores, además de la cantidad de tiempo de la prueba gratuita. Entonces se eligió como la mejor opción a GCP, el cual provee un servicio rápido con una prueba gratuita de 90 días, sin costo adicional alguno.

3.3 Implementación del contenedor de Base de Datos y alojamiento en la nube

Comenzando con algunos detalles y requisitos previos se constata que para poder realizar cualquier actividad dentro de GCP, con sus diferentes beneficios se necesita de una tarjeta de débito o crédito. Debido a que va a verificar que puede pagar cualquier servicio adicional fuera del saldo de $300 gratuitos. Se resalta que, la plataforma no va a cobrar ningún monto adicional en los 90 días de prueba gratuita; dentro de la misma es importante verificar la facturación en cualquier proyecto o actividad que se haga. Resaltado lo anterior se puede ingresar de manera fácil a la pantalla principal de Google *Cloud,* donde se podrá a su vez trabajar en el proyecto de titulación, como se logra mostrar en la Figura 3.1 [21].

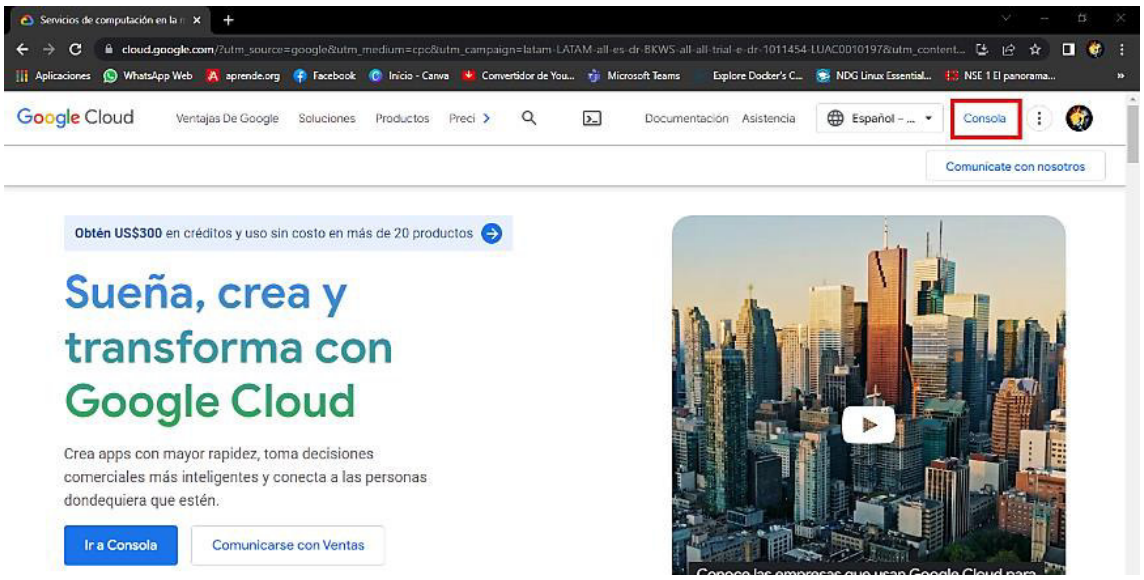

Figura 3.1 Vista para la entrada a GCP una vez registrado

Ya dentro de la plataforma en su menú principal se va a crear un nuevo proyecto, se deberá ir a la parte superior izquierda cerca del menú de navegación para seleccionar

un proyecto, donde se dará clic el botón de “Proyecto Nuevo”. Aquí se presentará un menú con los proyectos que se han creado recientemente con su identificador o ID, dónde en este caso se va a agregar un proyecto denominado “ProyectoT”, como se observa en la Figura 3.2. Se debe ubicar dentro del mismo para desarrollar el presente proyecto de titulación, puesto que es de gran relevancia para usar el intérprete de comandos o la interfaz de línea de comandos (CLI) de GCP [5].

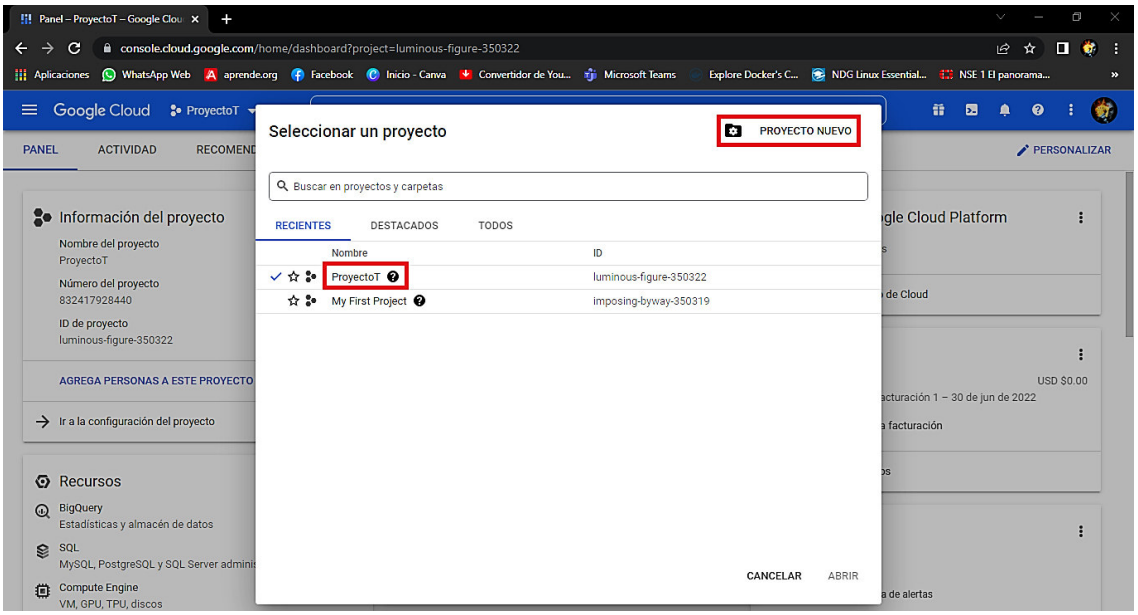

Figura 3.2 Creación del proyecto nuevo denominado “ProyectoT”

Una vez creado el proyecto en el que se va a trabajar, en la página principal se tendrá que ingresar al menú de navegación que está localizado en el lado superior izquierdo, donde se encuentran las diferentes herramientas las cuales se pueden usar según los requerimientos del usuario. Acotando con un dato importante es que en función de las herramientas utilizadas se va a descontar el saldo que da la plataforma [5].

Dejando lo anterior claro se procede con la implementación de una base de datos dentro de esta plataforma, primero se selecciona el servicio que dispone GCP para el registro de contenedores llamado *Container Registry* donde se procede a seleccionar la sub pestaña de imágenes, como se resalta en la Figura 3.3. Cabe recalcar que, si no está activado este servicio, se lo debe activar, esto se demorará aproximadamente entre 1 a 3 minutos máximo [5].

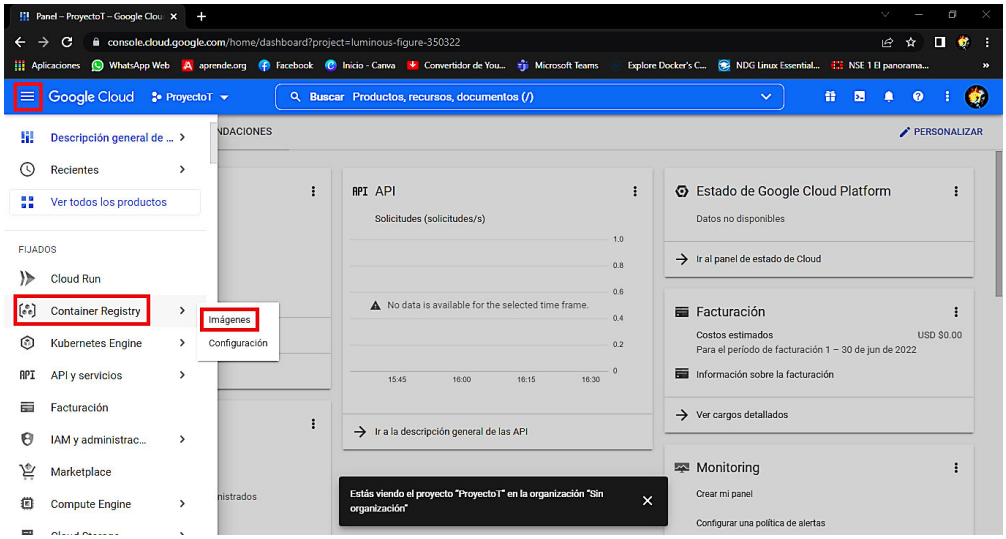

Figura 3.3 Vista del menú de navegación con la pestaña *Container Registry*

Una vez activado e ingresado a este servicio se podrá ver como título principal "Repositorios" donde se va a tener una tabla con el Nombre, el Nombre de Host y la Visibilidad de cada directorio creado. Para manipular este servicio de una mejor manera se abre el CLI de esta plataforma, que está ubicado en el lado superior derecho de la pantalla, como se visualiza en la Figura 3.4 [5].

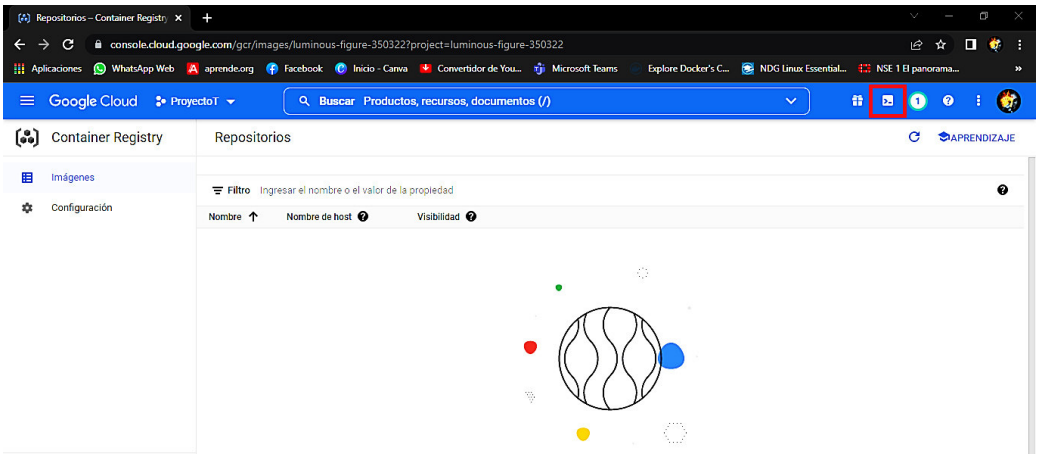

Figura 3.4 Vista de la pestaña *Container Registry* y ubicación del CLI de GCP

Una vez abierto el intérprete de comandos de GCP se debe tener ya fijada la imagen para el contenedor de base de datos, debido a que este se lo va a implementar dentro de esta plataforma. Por lo tanto, se debe dirigir a *Docker Hub* la cual es una plataforma gratuita y libre que maneja los repositorios de contenedores de la comunidad, como se muestra en la Figura 3.5. Donde permite albergar varias imágenes las cuales pueden tener diversas categorías de aplicaciones, servicios, seguridad, entre otras. Además, tienen diferentes etiquetas como Imagen Oficial de *Docker*, *Plugins*, Verificación de Editor y Programa de Código Abierto [25].

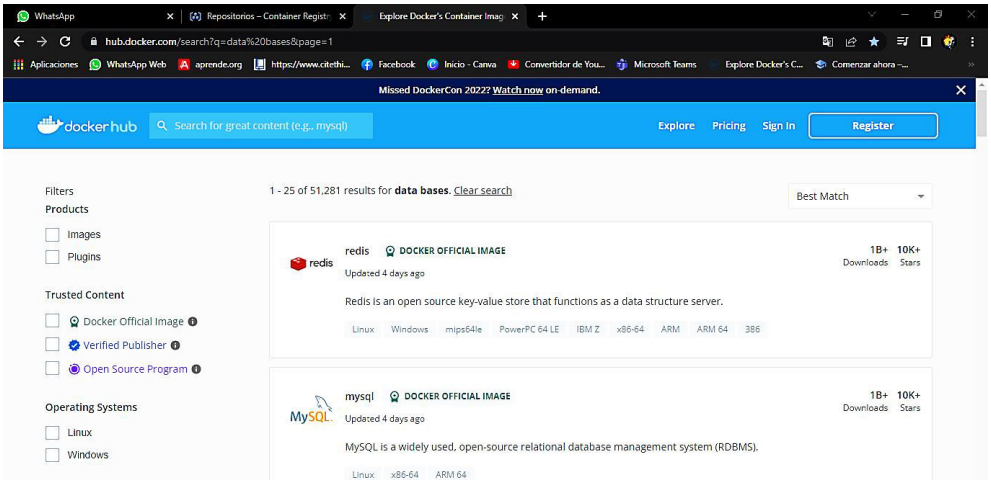

Figura 3.5 Vista de la página principal de *Docker Hub*

Para simplificar la búsqueda se tomó en base los SGBD dentro de la plataforma de *Docker Hub* que sean imágenes oficiales. Con lo cual se pudo filtrar los diferentes SGBDs más populares que cuentan con plantillas para poder usarlas en un contenedor. Se elige al gestor de base de datos RethinkDB, ya que tiene una plantilla oficial dentro de la plataforma de *Docker Hub,* la cual va a permitir que sea más factible la implementación de este gestor de base de datos como contenedor, como se puede ver en la Figura 3.6.

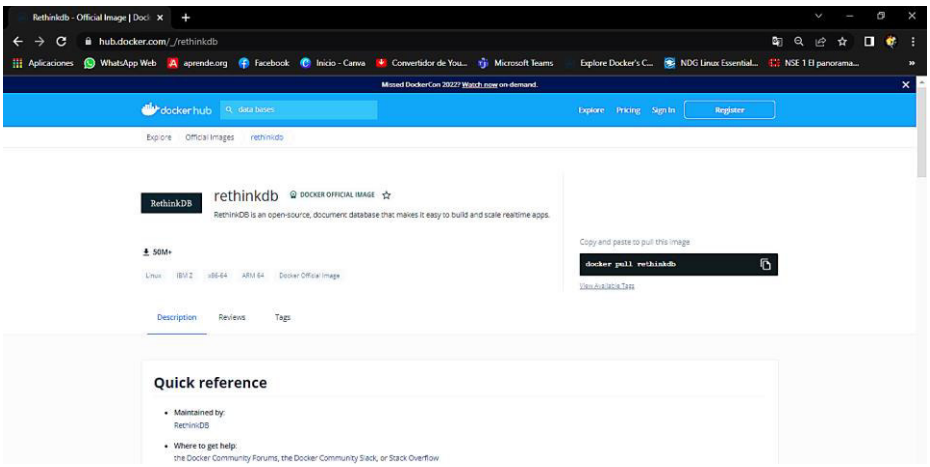

Figura 3.6 Imagen del SGBD de RethinkDB

Para traer la imagen del contenedor de RethinkDB dentro del CLI de GCP se usa el comando ***docker pull***. Este comando tiene la finalidad de descargar cualquier imagen del registro de contenedores de GCP o del repositorio de *Docker Hub,* teniendo como prioridad principal *Container Registry*. Para observar las imágenes que se han descargado desde el repositorio de *Docker* se usa el comando ***docker images,*** el cual va a lanzar información como el Repositorio, Etiqueta, Identificador, Fecha de Creación y Tamaño como se logra mostrar en la Figura 3.7 [5].

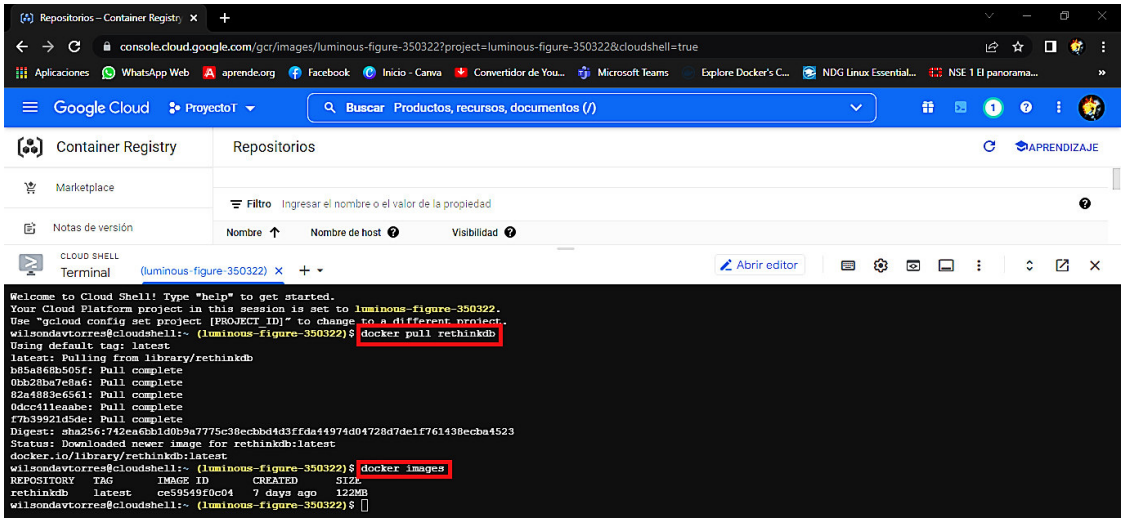

Figura 3.7 Ejecución del comando para descargar la imagen y para su verificación

Se debe tener en cuenta que el intérprete de comandos solo va a estar habilitado y funcional durante un período de tiempo específico, y una vez culminado este tiempo se borrará todo lo que se hizo anteriormente. Dando por hecho que todas las actividades que se realizaron serán eliminadas, y para su resolución se puede trabajar dentro de Kubernetes de GCP [5].

Después de haber descargado la imagen del contenedor de base de datos, se va a crear e implementar el contenedor. Aquí se debe recalcar que, si la imagen de la BD no funciona al ejecutarlo dentro del CLI, no va a poder implementarse en el GCP. Todos los comandos que se involucran en esta sección del documento tienen base y procedencia de *Docker* [26].

Como se aprecia en la Figura 3.8 el comando para crear el contenedor es complejo, por lo que a continuación, se va a explicar las diferentes partes del mismo:

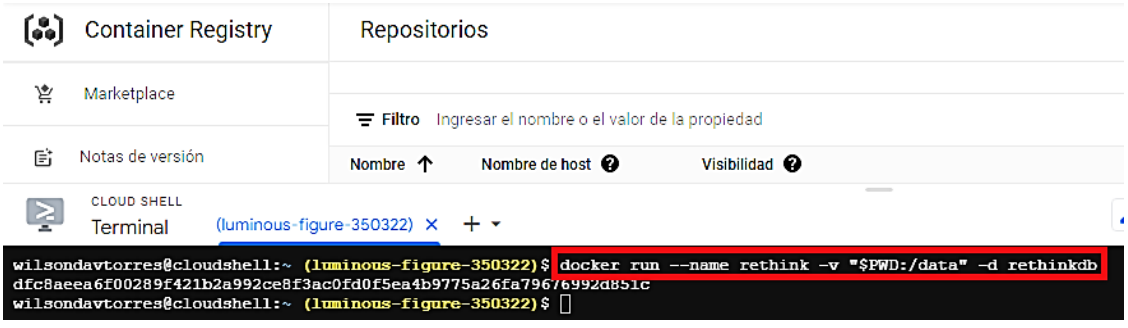

Figura 3.8 Ejecución y vista del comando para implementar el contenedor

docker run: Este comando permite correr al contenedor en base de la imagen, con su respectivo ID único del contenedor lanzado en pantalla [26].

--name: Es el nombre del contenedor que se le va a dar, dependiendo de lo que se vaya hacer, y en este caso será “rethink” [26].

-v "$PWD:/data": Especifica dentro de la configuración de volúmenes la ubicación donde estará el directorio actual de trabajo (todo lo que está entre comillas), a partir de donde se ejecuta el comando [27].

-d rethinkdb: Correspondiente a la última parte del comando, donde **-d** hace alusión que al contenedor se lo correrá en segundo plano, mientras que lo siguiente especifica la imagen de RethinkDB que se usará [26].

Una vez realizado lo anterior, el contenedor deberá estar corriendo, para poder verificarlo se tienen los comandos ***docker ps - a*** y ***docker ps.*** El primero específica todos los contenedores creados sin ninguna clase de distinción en fecha de creación o su tiempo de ejecución con los diferentes datos de: ID del contenedor, ID de la imagen, Comando, fecha de creación, estatus, puertos y nombres. Mientras que ***docker ps***, solo muestra los contenedores (con la misma información anterior) que están actualmente encendidos o corriendo. Cualquiera de los dos comandos da como resultado que el contenedor está corriendo exitosamente, como se logra ver en la Figura 3.9 [5].

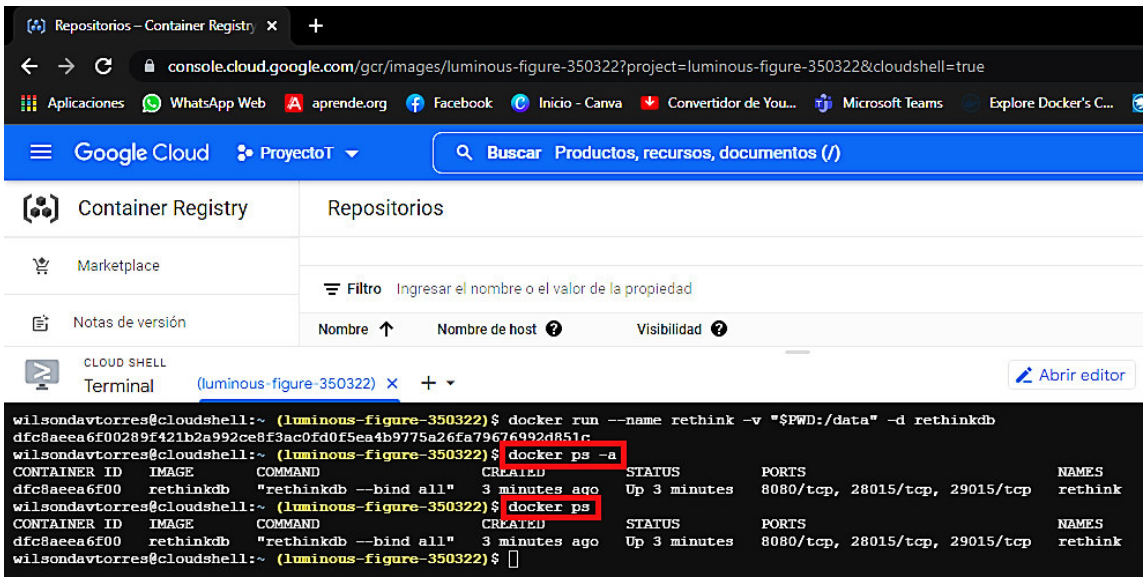

Figura 3.9 Ejecución de los comandos para comprobación del estado de arranque del contenedor

Tras la verificación exitosa de la implementación del contenedor mediante comandos dentro de GCP, se va a cambiar de repositorio en donde se lo va a alojar. Se utiliza el comando ***docker tag*** el cual da la facilidad de poder cambiar el repositorio, y tiene relación con el identificador o ID de la imagen que es un número único. Dentro de la estructura de este comando se coloca ***docker tag*** seguido del identificador único de la

imagen del contenedor y de un espacio. Luego viene el gcr.io el cual está pensado para dar una ubicación de dónde se pueda alojar a la imagen del contenedor, como se lo puede visualizar en la Figura 3.10 [5].

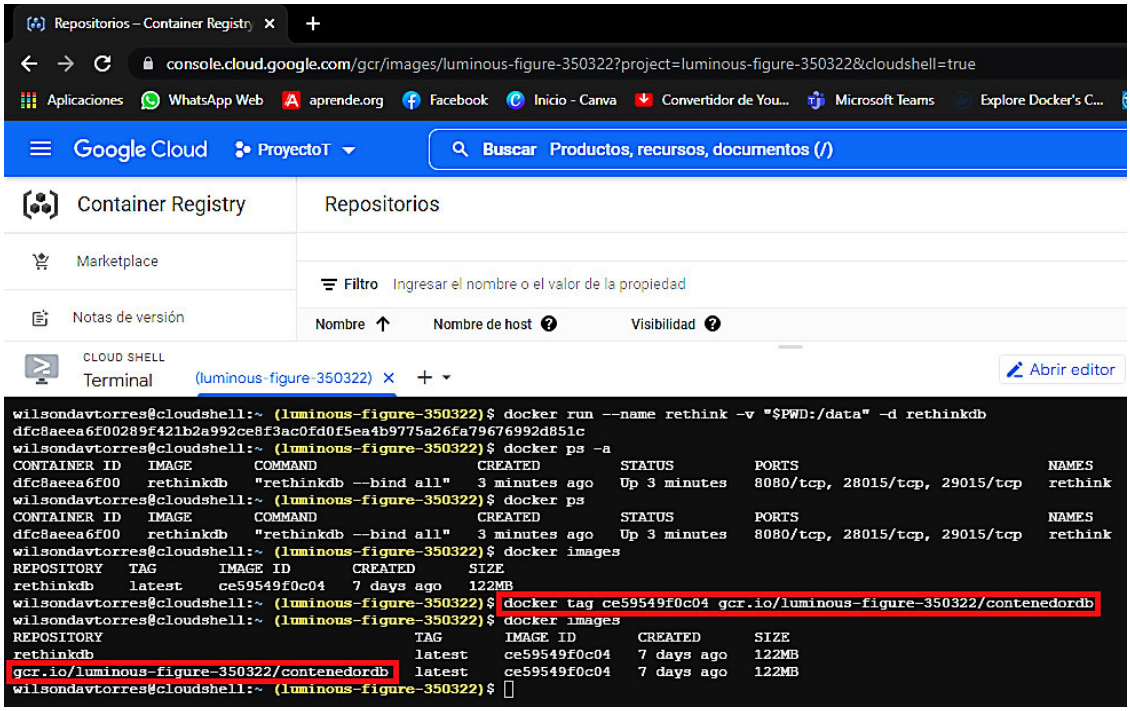

Figura 3.10 Ejecución del comando para la asignación de repositorio de la imagen

Existen tres clases de ubicaciones, la primera es gcr.io que corresponde a Estados Unidos, us.gcr.io equivalentemente su locación es en Estados Unidos pero es autónomo del anterior. El segundo de eu.gcr.io que corresponde a la Unión Europea y finalmente Asia que es asia.gcr.io. Se eligió a gcr.io debido a que su tiempo de respuesta en retardo es más conveniente. Luego le sigue un *slash* junto con la identificación única del proyecto, el cual se lo puede encontrar eligiendo el nombre del proyecto y copiando este requerimiento importante [5].

Después del siguiente *slash* se define el nombre del directorio el cual va a albergar este contenedor, y para este caso se va a llamar “contenedordb” (si es necesario una etiqueta al directorio se la puede agregar con “:” y la etiqueta). Últimamente se procede a colocar la imagen dentro del registro de contenedores en GCP, donde se utiliza el comando ***docker push*** que va seguido del nuevo repositorio creado, lo que se menciona en el comando anterior (después del identificador de la imagen en su totalidad), como se observa en la Figura 3.11. El tiempo de ejecución del comando es relativamente corto, aunque durante el proceso puede exigir ciertos permisos para poder correrlos los cuales se los otorga instantáneamente, como se puede ver en la Figura 3.12 [5].

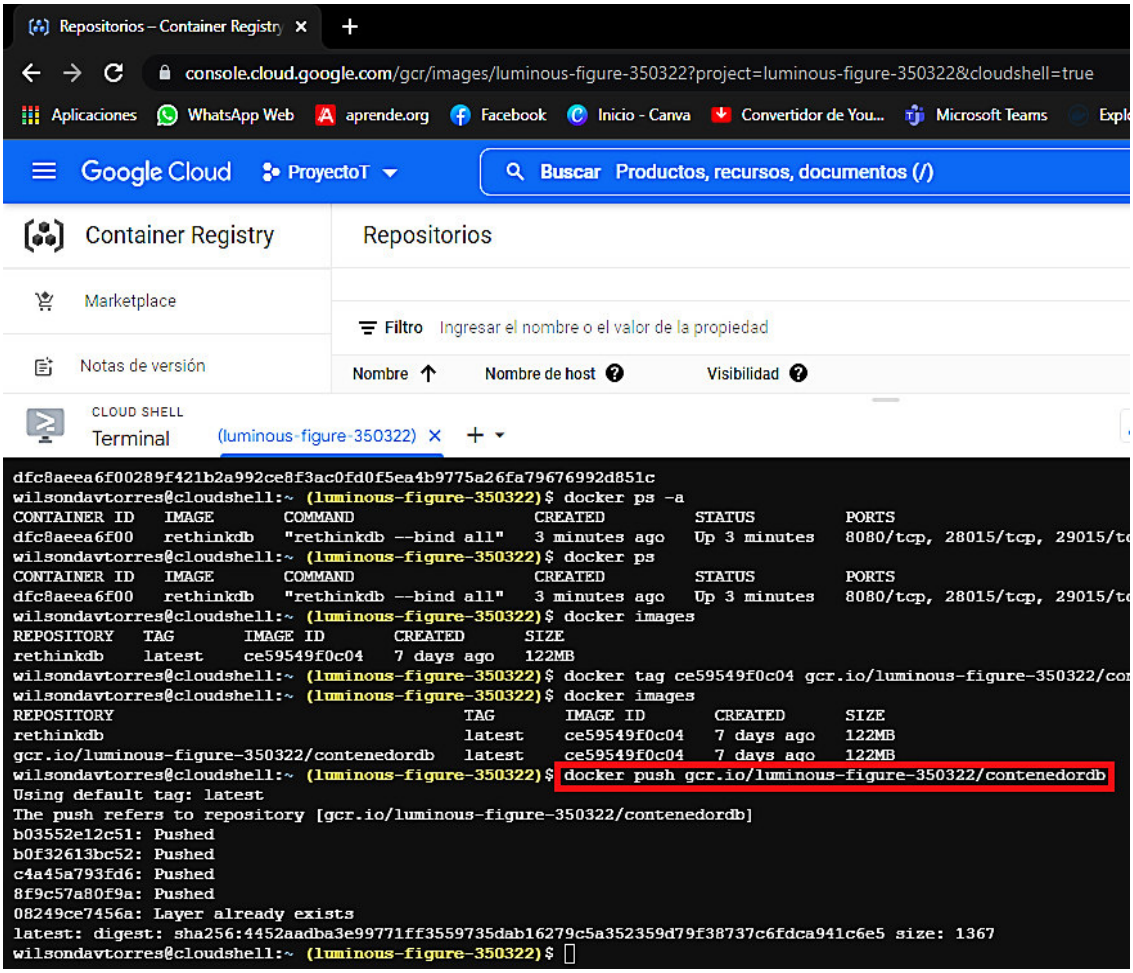

Figura 3.11 Aplicación del comando ***docker push***

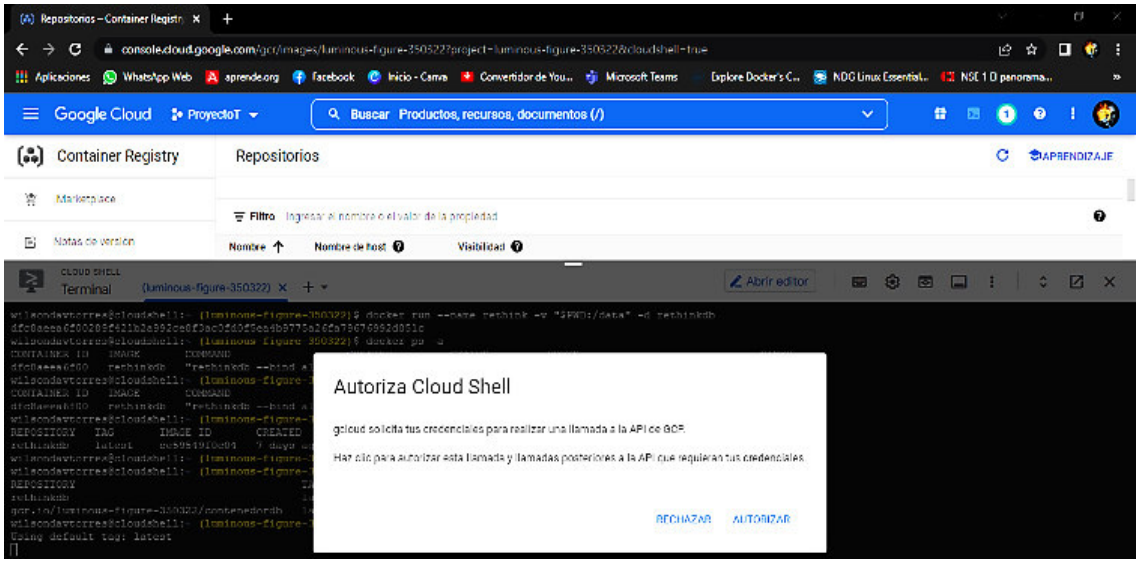

Figura 3.12 Vista de permisos adicionales para subir el directorio del contenedor

Para poder verificar que se lo hizo exitosamente, se recarga nuevamente la página donde deberá aparecer el directorio o carpeta denominada “contenedordb”, en la cual

va a estar alojada la imagen previamente dicha, como se logra observar en la Figura 3.13. Al ingresar se puede ver muchos detalles como tipo de imagen, proyecto, repositorio, tamaño, fecha de creación, etiquetas, entre otros como se ve en la Figura 3.14 [5].

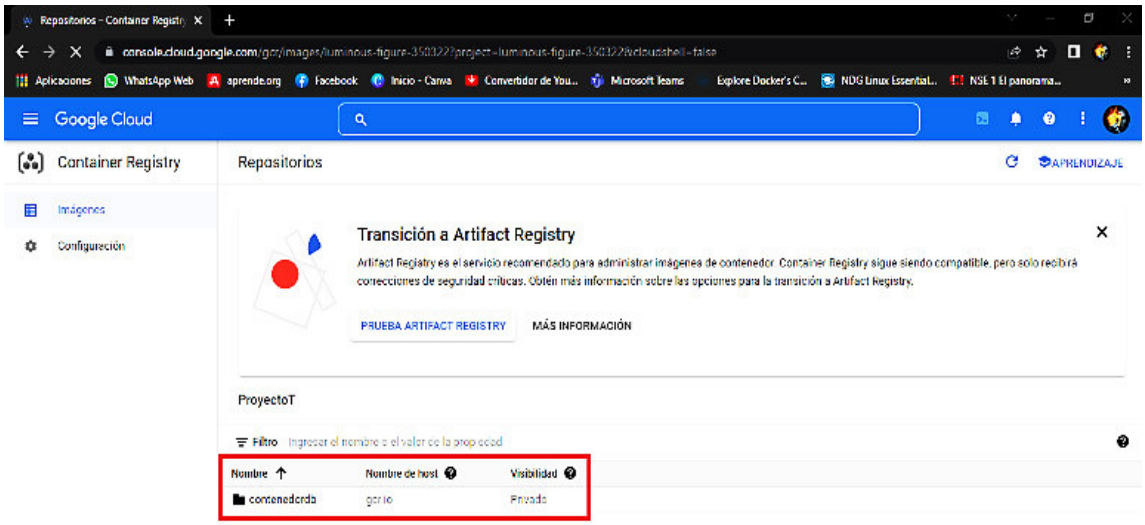

Figura 3.13 Comprobación del directorio con la imagen albergada en el repositorio

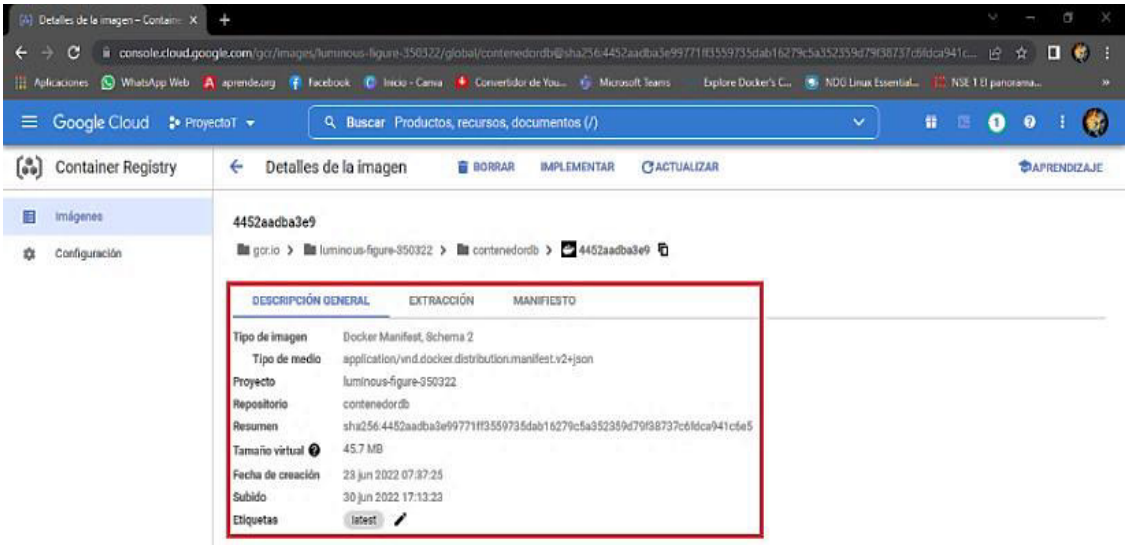

Figura 3.14 Información de la imagen albergada en el directorio

Una vez hecho lo anterior de manera exitosa llega la parte de la implementación del contenedor como tal. Se procede a seleccionar dentro de la carpeta del contenedor la alternativa que se localiza en el lado superior derecho denominada “IMPLEMENTAR”, la cual da tres alternativas de implementación de *Cloud Run,* Google *Compute Engine* (GCE) y Google *Kubernetes Engine* (GKE) como se puede ver en la Figura 3.15. Para la implementación de este contenedor se seleccionó *Cloud Run*, que da la facultad de manipular, escalar y verificar el SGBD con un precio de implementación mínimo [5].

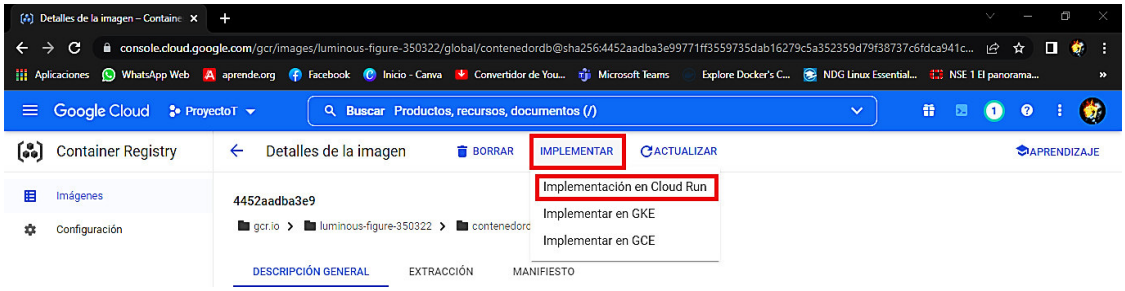

Figura 3.15 Vista del botón “IMPLEMENTAR” con sus opciones disponibles

Después de seleccionar la opción anterior, se procede a verificar la ubicación de la imagen que anteriormente se subió al *Container Registry.* Luego se debe dar un nombre al servicio para que se lo pueda implementar, en este caso va a ser “serviciodb”, como se ve en la Figura 3.16. Dentro de la pestaña denominada “Precios y asignación de CPU” se encuentran dos opciones, donde la primera da la posibilidad de asignar la CPU solamente cuando se reciba una solicitud, la cual define que solo va a asignar los recursos durante el procesamiento de una sola solicitud. La segunda opción que menciona que la CPU siempre asignada, da capacidad de dar el servicio que se encuentre en constante ejecución en un puerto fijo, donde en el caso de una base de datos es la más idónea. Sin embargo, se debe tener en cuenta que para esta opción el cobro por mantenerlo aumenta. También existe una pestaña de “Ajuste de escala asignada” el cual menciona el número de instancias que llegarán al servicio que por defecto se le va a dejar de 0 a 100, como se ve en la Figura 3.17 [5].

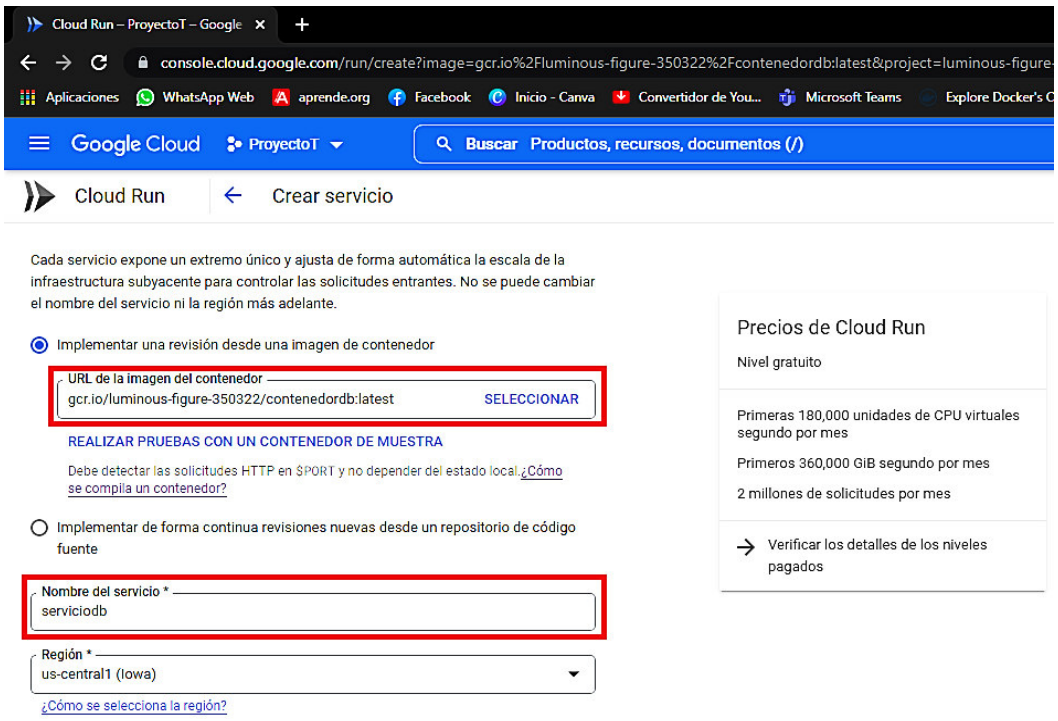

Figura 3.16 Verificación y asignación de nombre al servicio denominado “serviciodb”

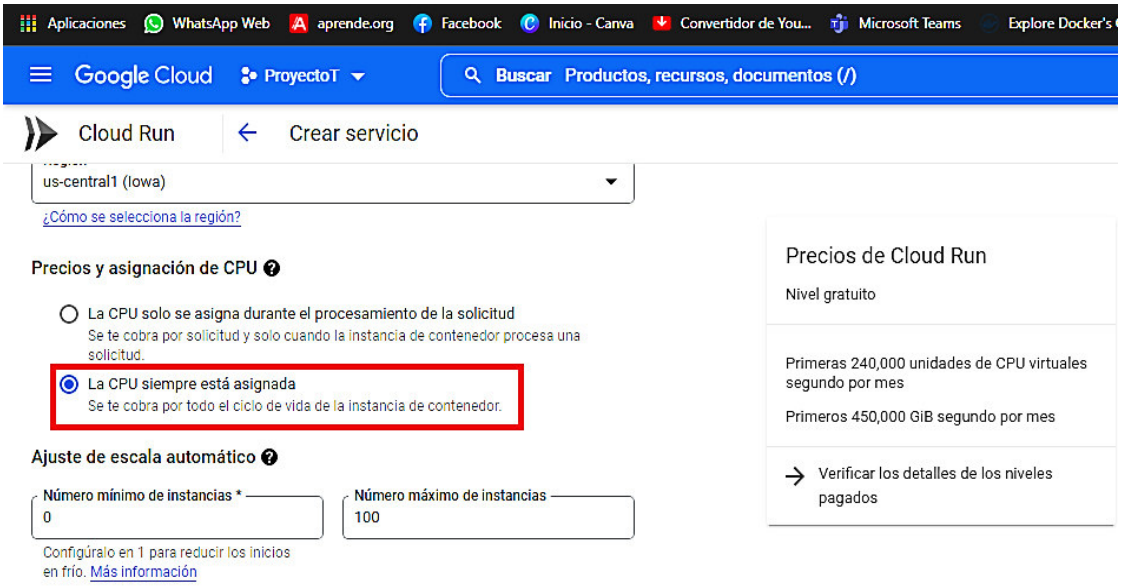

Figura 3.17 Pestaña de “Precios y asignación de CPU” y “Ajuste de escala asignada”

A continuación, dentro de las pestañas de “Entrada” se encuentra tres opciones para la entrada de tráfico; la primera es “Permitir todo el tráfico”, la segunda es “Permitir el tráfico interno y el proveniente de *Cloud Load Balancing”* y la tercera es “Permitir solo el tráfico interno”. Únicamente se va a seleccionar la primera la cual sale por defecto. Luego dentro de la pestaña de “Autenticación” donde se tienen dos opciones las cuales son permitir invocaciones sin autenticar y la segunda que requiere autenticación de manera predeterminada, donde se dejó la primera opción la cual permite las invocaciones sin autenticar, ya que es la más recomendable, como se muestra en la Figura 3.18.

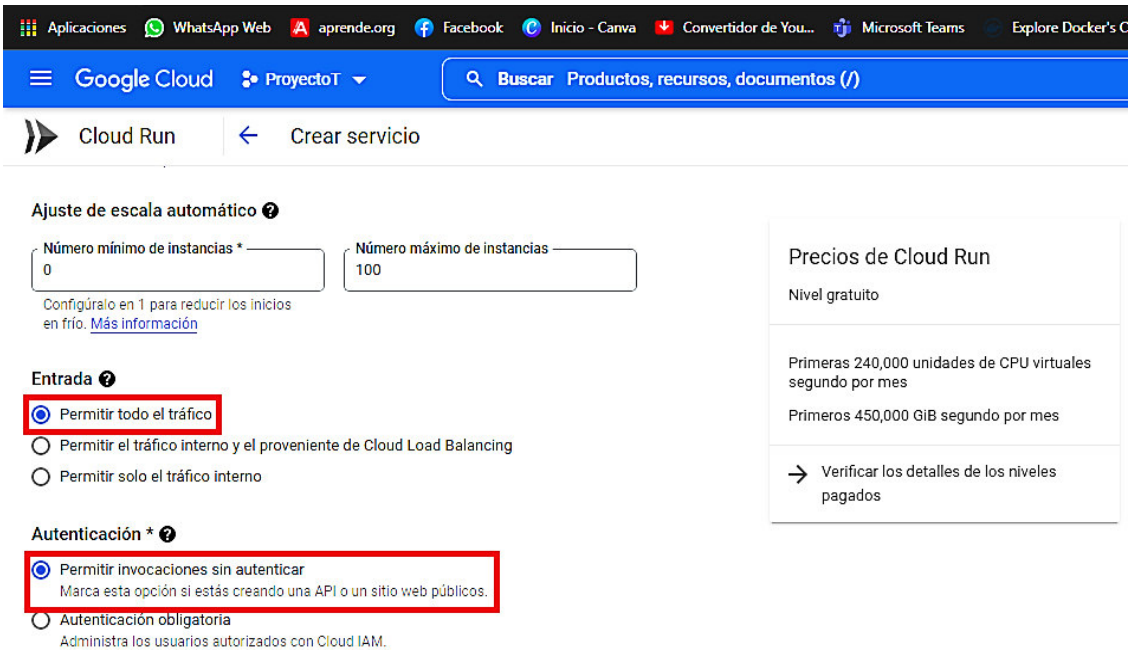

Figura 3.18 Vista de las pestañas “Entrada” y “Autenticación”

Existe una pestaña con opciones avanzadas llamada “Contenedores, variables y secretos, conexiones, seguridad” que permite definir el puerto, el número de procesadores, memoria, entre otros requerimientos. Sin embargo, debido a que esta base de datos es dirigida a la *web* se le va a dejar por defecto, porque funciona en el puerto 8080, como se logra observar en la Figura 3.19 [5].

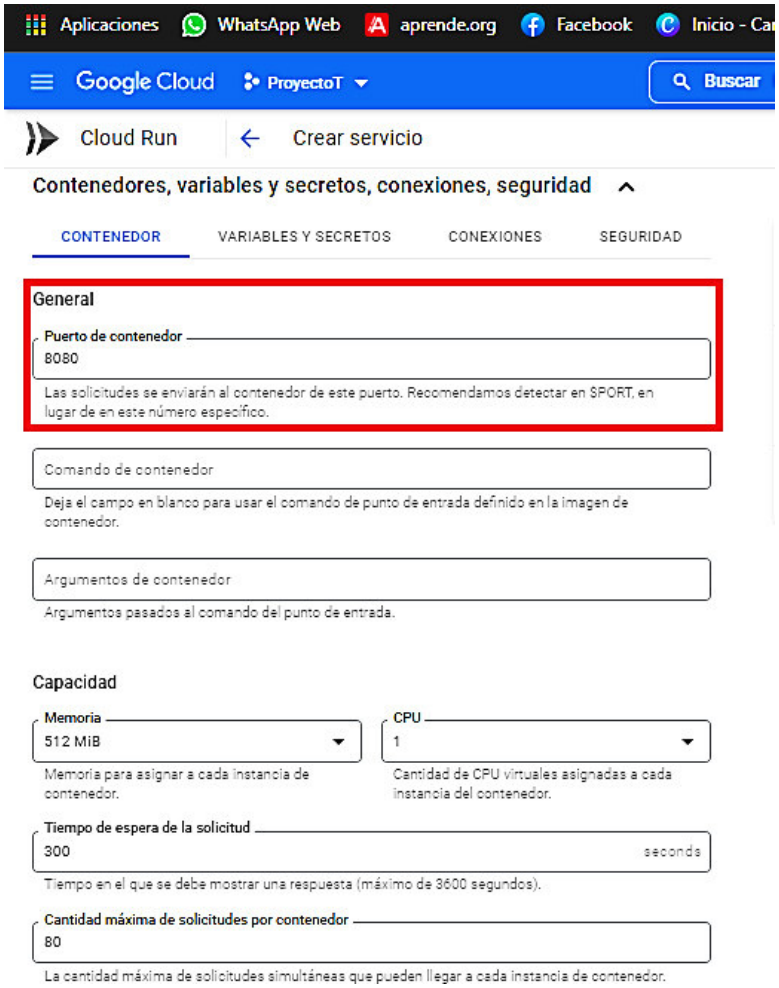

Figura 3.19 Vista de la pestaña con las opciones avanzadas por defecto

Después se le dio clic en el botón “CREAR” que está en la parte final izquierda de todo este proceso, lo cual da la posibilidad de crear el servicio dentro de GCP. Con esto la plataforma comenzó a configurar las políticas de *Identity Access Management* (IAM) los cuales permiten visibilidad y control de acceso de las instancias que se encuentran en la nube. Luego la plataforma examinará que la imagen durante el proceso no tenga algún tipo de error, se observa en Creando revisión, donde al encontrarlo va a parar la implementación y dará un mensaje para corrección (los más comunes son errores en el puerto y un error relacionado con la ejecución de la imagen). Finalmente da el servicio de Enrutando el tráfico, lo cual depende de lo que se haya seleccionado en la opción de Entrada indicada en la Figura 3.18, todo esto queda corroborado en la Figura 3.20 [5].

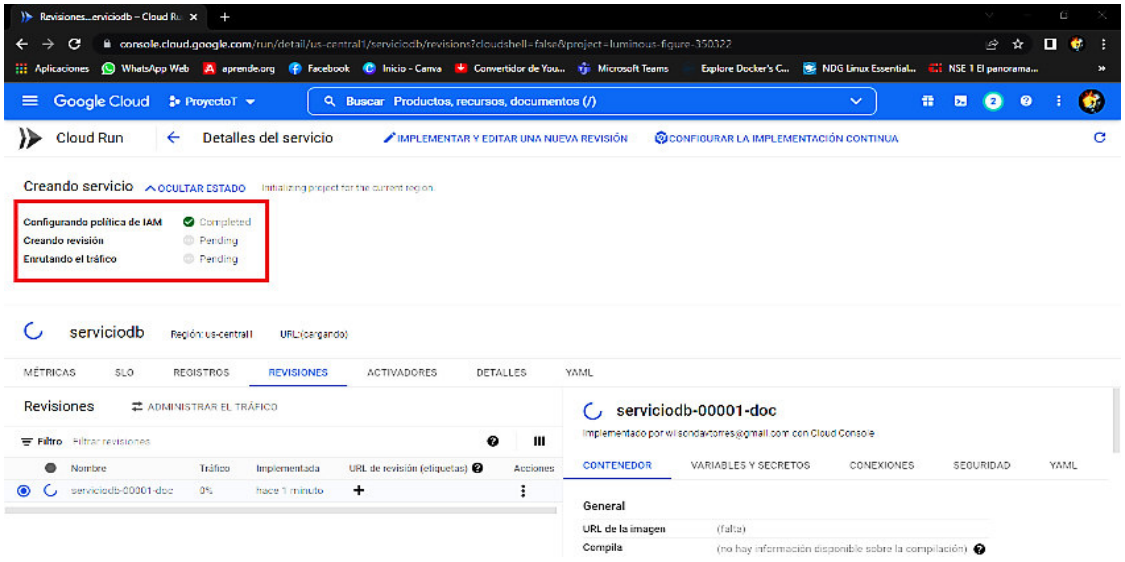

Figura 3.20 Proceso de creación del servicio de SGBD en GCP

Después de la culminación de la revisión e implementación de manera idónea de este servicio como tal, GCP proporciona un *link* o enlace resultante, el cual otorgará el acceso al SGBD de RethinkDB, como se observa en la Figura 3.21.

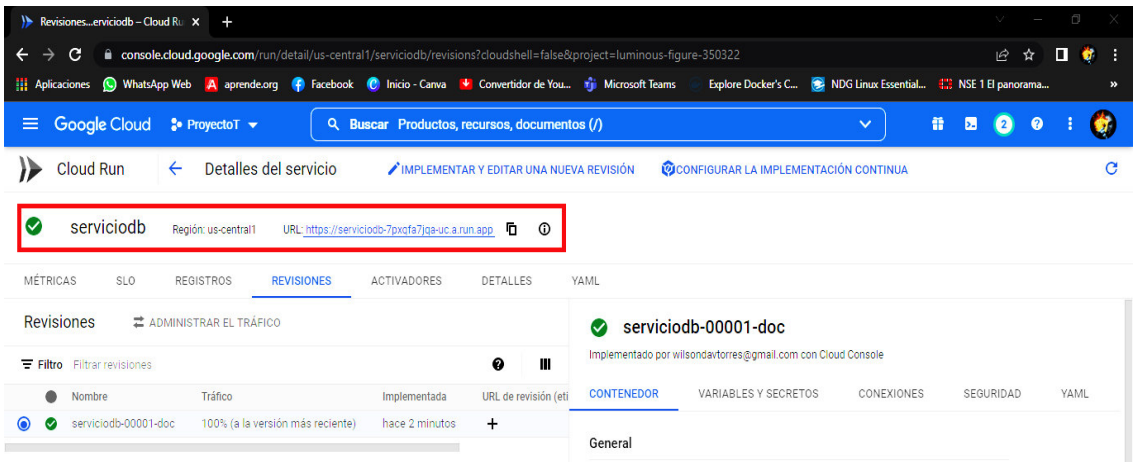

Figura 3.21 Creación del enlace para el acceso a la BD

3.4 Pruebas de funcionamiento y verificación de resultados obtenidos

Para constatar el funcionamiento del contenedor de la RethinkDB en el proveedor de servicios en la nube GCP, se usó el servicio de *Cloud Run*, junto con el CLI que proporciona la plataforma y el uso de *Docker* para poder correr la BD en base a la imagen de su contenedor.

Para corroborar su correcto funcionamiento se necesita dar clic sobre el *link* el cual permite ingresar al SGBD de RethinkDB, según se observa en la Figura 3.22.

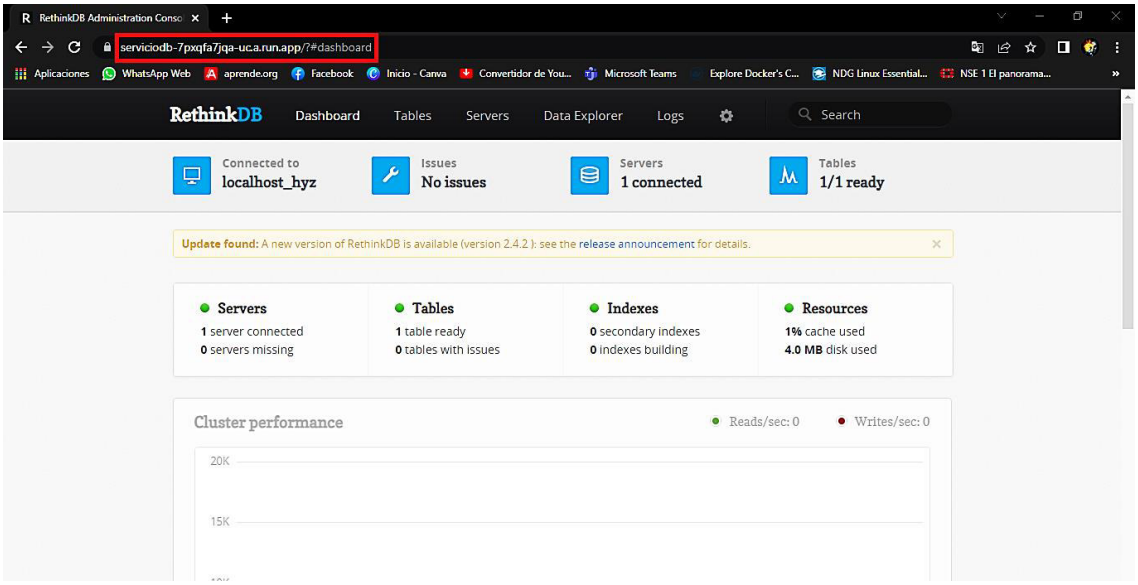

Figura 3.22 Verificación visual del SGBD de RethinkDB funcionando

Para constatar que el enlace funciona en cualquier computadora, se procedió a ejecutarlo en tres redes diferentes que poseen direcciones IP disímiles. En la primera red, hogar del autor, se va a crear una BD dentro de RethinkDB con tres registros dentro del clúster establecido. Para ello en la página *web* de la BD se dirige a la pestaña de "*Tables*", por defecto aparece una BD "*test*" la cual no se va a usar; se creó una BD mediante el botón "*Add Database*", que se localiza en el lado superior derecho, con el nombre de "Prueba".

Dentro de esta se creó una tabla, mediante el botón "*Add Table*" ubicado en la base de datos, y para este caso se lo llamó "Registro", como se puede apreciar en la Figura 3.23.

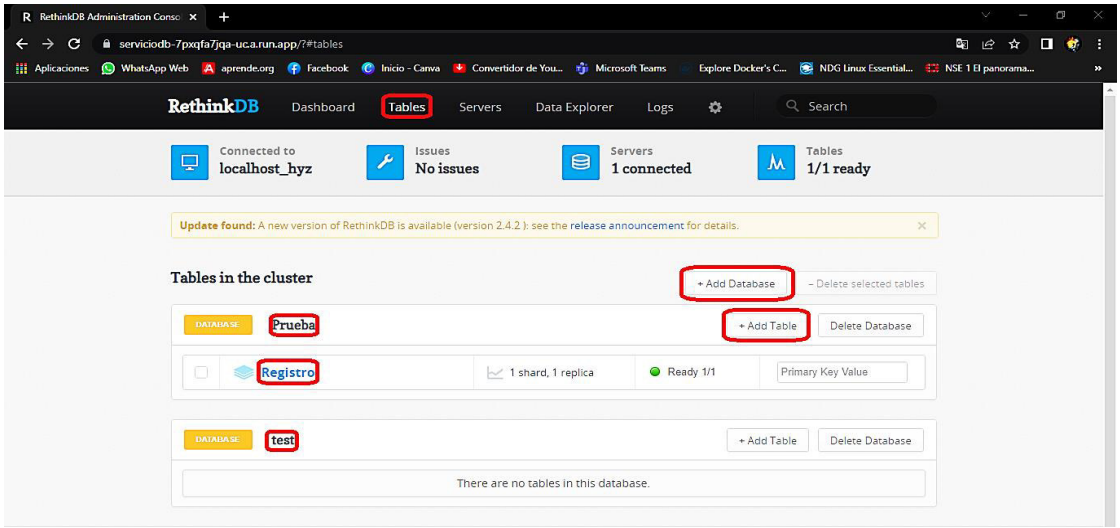

Figura 3.23 Creación de una BD llamada "Prueba" con su tabla "Registro"

Para el ingreso de registros dentro de la base de datos en la tabla, este SGBD tiene un lenguaje de consulta el cual se denomina ReQL (RethinkDB *Query Language*). Para poder ingresarlos dentro de esta plataforma, se debe dirigir a la pestaña que menciona "*Data Explorer*". Aquí se tienen botones importantes que son "*Clear*" que sirve para limpiar su intérprete de comandos y "*Run*" que sirve para correr un comando específico.

Primero se va a verificar cuantas BD se han creado mediante el comando **"r.dbList()"** y se le corre, apareció en la parte de abajo el resultado de las BD de "*test*", "*rethinkdb*" y la que se creó "Prueba", como se puede apreciar en la Figura 3.24 [28].

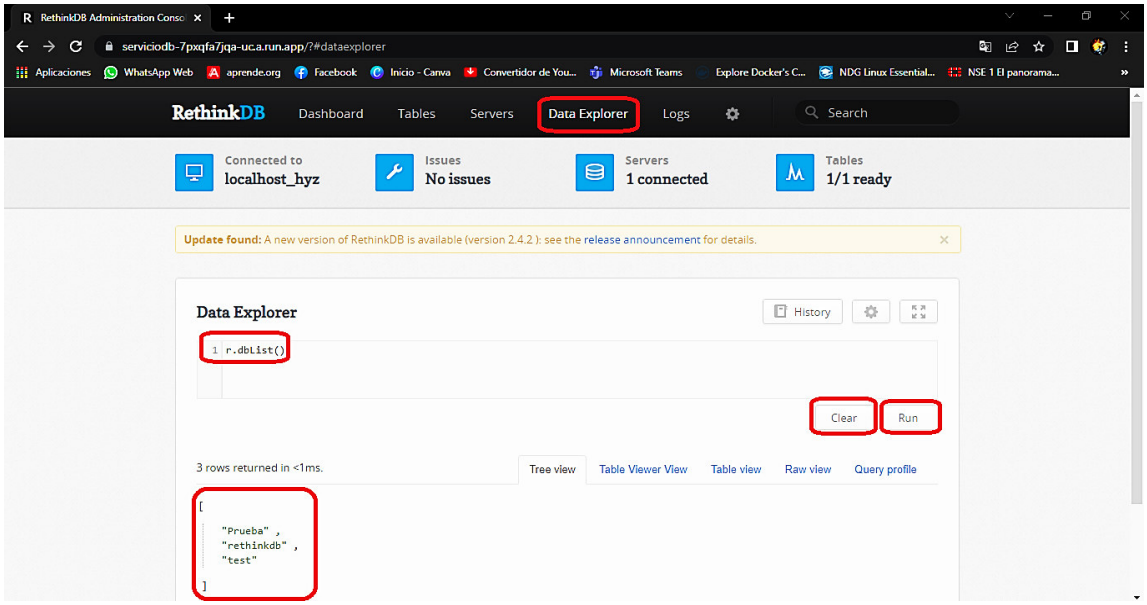

Figura 3.24 Vista de la ejecución del comando para ver las BD en la pestaña "*Data Explorer*"

Una vez verificado que la BD haya sido creada con su tabla, se va a ingresar los siguientes comandos que van a crear la matriz, la cual está compuesta de cuatro columnas por tres filas. La columna de "id" o identificación única en cada registro se crea por defecto (también se la puede agregar por comandos); además se agregan los campos de NOMBRE, SEXO y EDAD con sus registros; en la Figura 3.25 se observan los comandos los cuales son explicados a continuación [28].

En la primera línea el comando **"r.db('Prueba').table('Registro').insert(["** se refiere a detallar la ubicación de la BD que va a ser "Prueba", junto con el estamento de tabla que va a ser "Registro" (también sirve para ingresar nuevos registros a la tabla existente, en campos específicos). Mediante la siguiente parte que menciona "*insert*" junto con un paréntesis "()" y un corchete "[]", se agregó los registros de NOMBRE, SEXO y EDAD (**{NOMBRE: "Wilson Torres", SEXO: "M", EDAD: 22}**), abriendo unas llaves "{}".

Se va anexando la información particular donde, para separar cada registro después de las llaves va una coma. También si se quiere que una BD se cree, y que automáticamente se vuelva a actualizar, mostrando todos los cambios que se hayan realizado, se puede poner al final del comando punto y coma (separa el orden y ejecución de los comandos) y poner el comando "**r.db('Prueba').table('Registro')**" que sirve para actualizar y ver las modificaciones que se han hecho en la tabla [28].

Figura 3.25 Comandos para agregar registros a la tabla

Después de aplastar el botón "*Run*" se puede observar el árbol *Tree View* que se crea con los diferentes registros y sus campos llenos. También tiene otras opciones como la visualización de la tabla de manera estándar *Table Viewer View, Table View* que muestra la información con mejor visualización; se observa también *Raw View* donde los registros se revelan en forma de comandos y finalmente el perfil de consulta *Query Profile.* Todo esto se puede apreciar en la Figura 3.26 y Figura 3.27 [28].

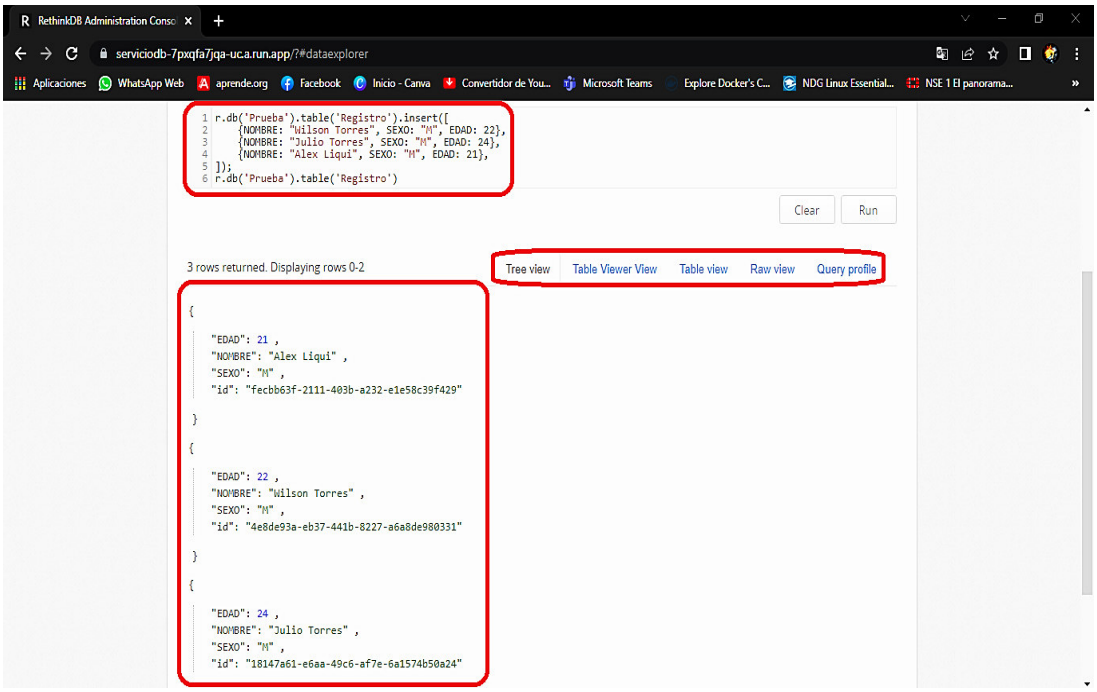

Figura 3.26 Vista del Árbol de los registros creados

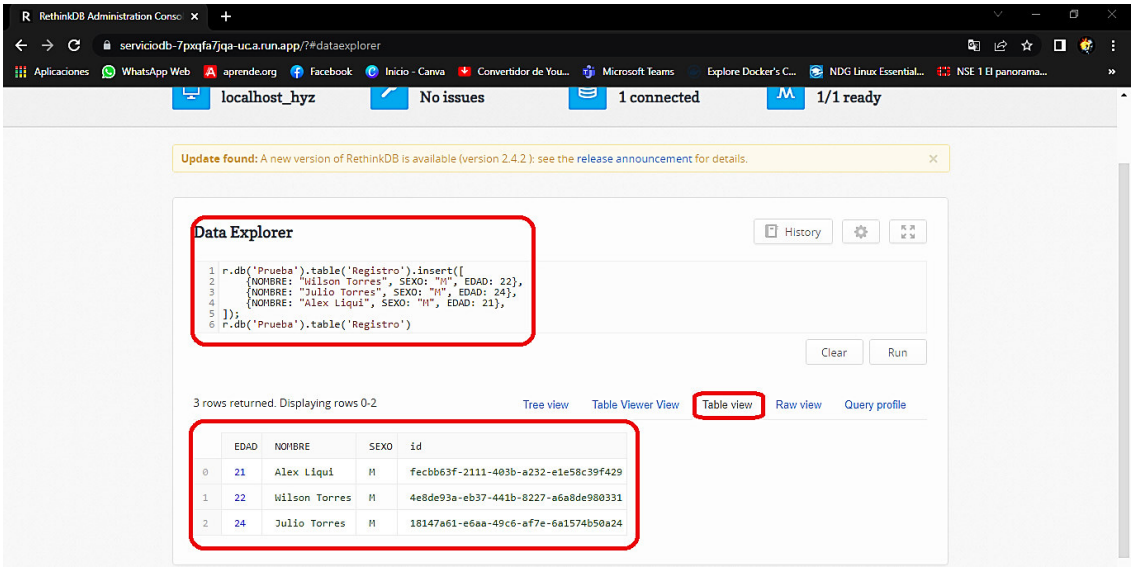

Figura 3.27 Vista del de la Tabla específica de los registros creados

Para la verificación con más detalles de la BD se va a dirigir a la pestaña de *Tables*, seleccionando la tabla creada. Una vez ingresado se ve las estadísticas de lo que se ha escrito o leído, donde para explorar más se puede ir a la parte superior derecha en el botón "*Explore*" (como se puede apreciar en la Figura 3.28) la cual lleva a la parte inferior de la pantalla. Ahí es donde se muestra la tabla creada con todos los datos ingresados de la id, EDAD, NOMBRE y SEXO que se colocó para crear la BD. Esto se aprecia en la Figura 3.29 junto con la red específica donde se realizó esta primera verificación.

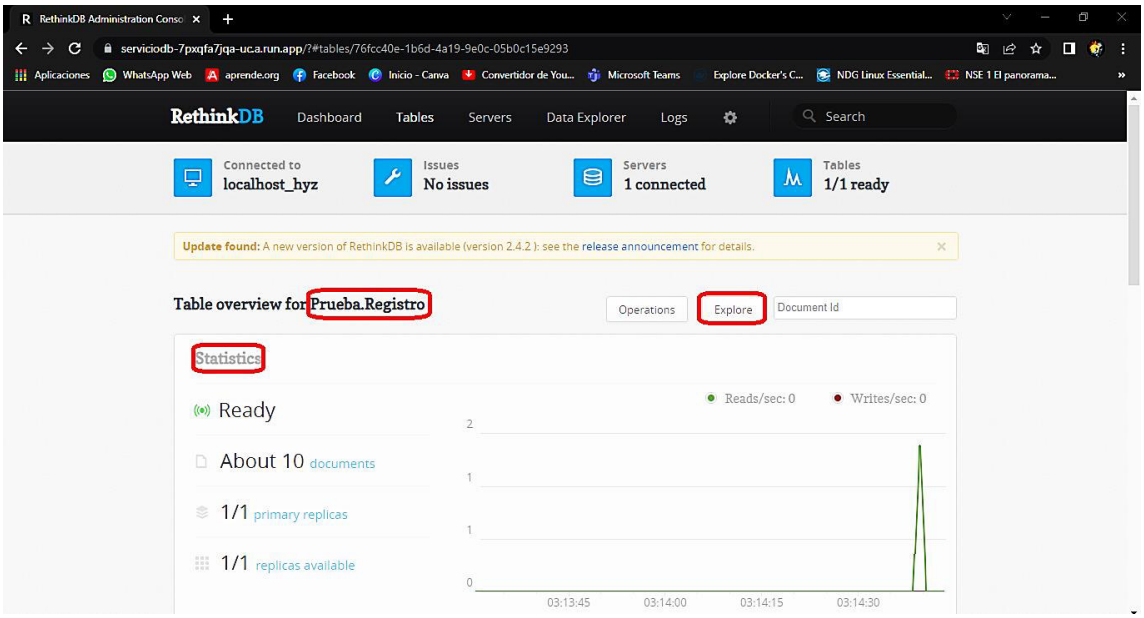

Figura 3.28 Vista de las estadísticas dentro de la BD de "Pruebas"

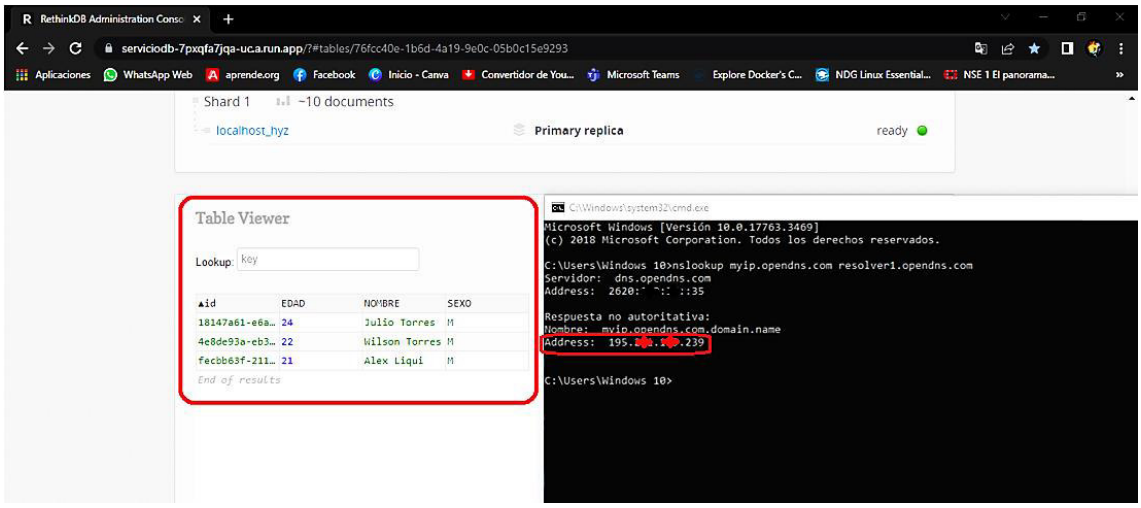

Figura 3.29 Vista de la tabla de forma específica dentro de la BD junto con la primera red

Mediante otra red, otro espacio físico, se realiza una nueva verificación. Se indexó un registro nuevo a la base de datos “Prueba”; en la Figura 3.30 se observa en *Data Explorer* los comandos para la indexación del nuevo registro de “Mayerly Chamorro”, en la parte inferior se observa la vista del árbol formado; se verifica también el direccionamiento de esta red.

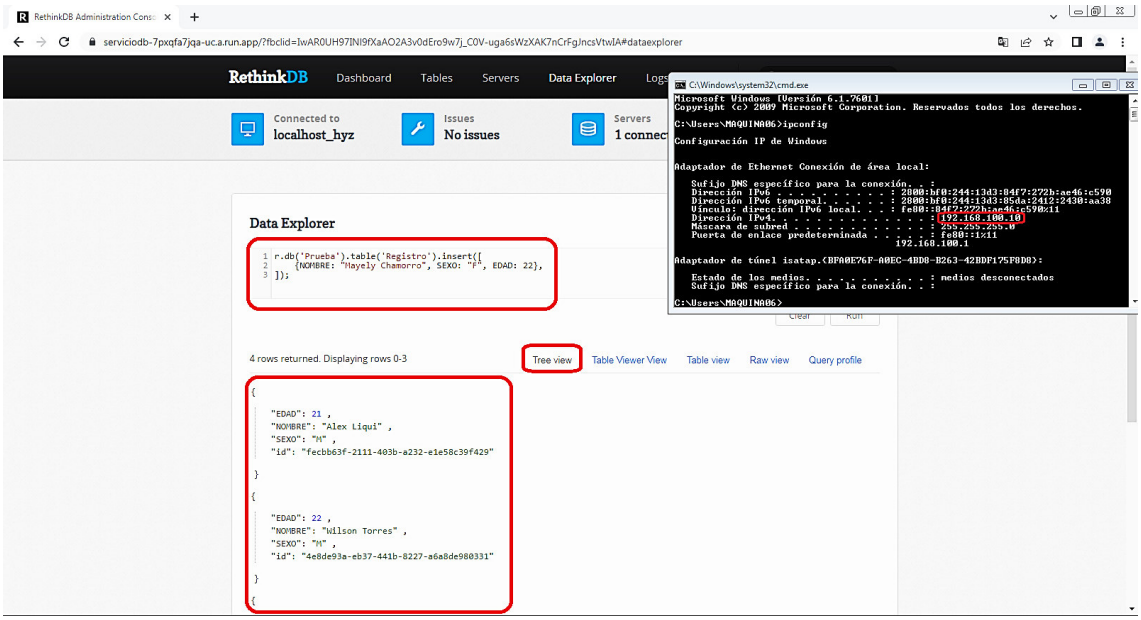

Figura 3.30 Vista de los comandos de la BD junto con la segunda red

En la Figura 3.31 dentro de la pestaña “*Tables*” en la base de datos “Prueba”, dando clic al botón “*Explore*” en las estadísticas, se observa la nueva indexación del nuevo registro de “Mayerly Chamorro” realizado en la tabla dentro de la segunda red, se verifica además el direccionamiento respectivo.

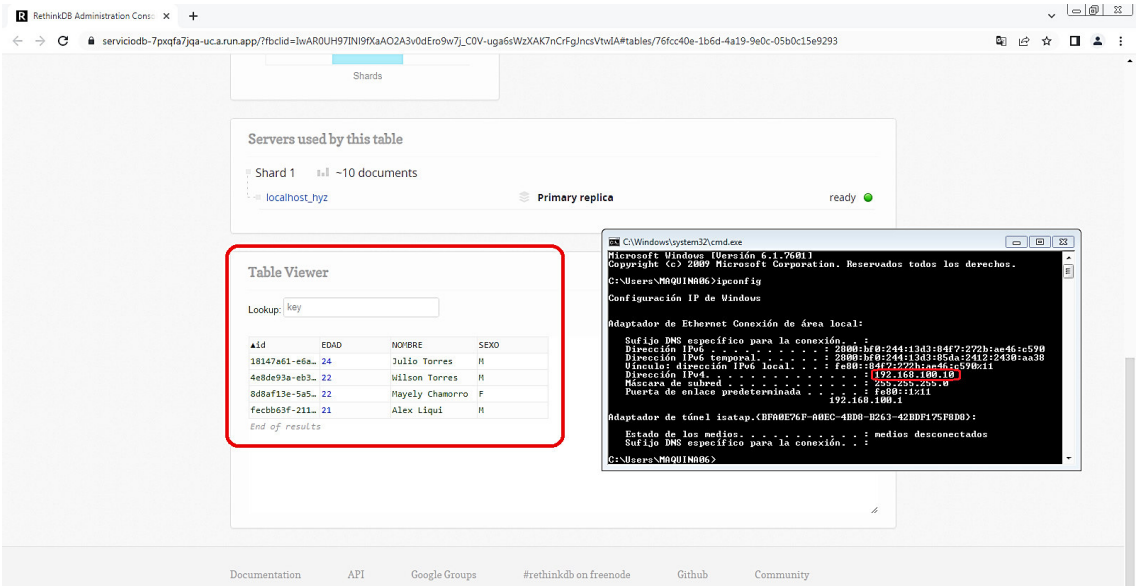

Figura 3.31 Vista de la tabla dentro de la BD junto con la segunda red

Para corroborar dentro de otra red diferente a la anterior, en otro espacio físico, se realiza una nueva verificación. Se indexó dos registros nuevos a la base de datos “Prueba”; en la Figura 3.32 se observa en *Data Explorer* los comandos para la indexación de los nuevos registros de “Cindy Morales” y “Taty Granda”, en la parte inferior se observa la vista de la tabla de forma amplia; se colocó también el direccionamiento de esta red.

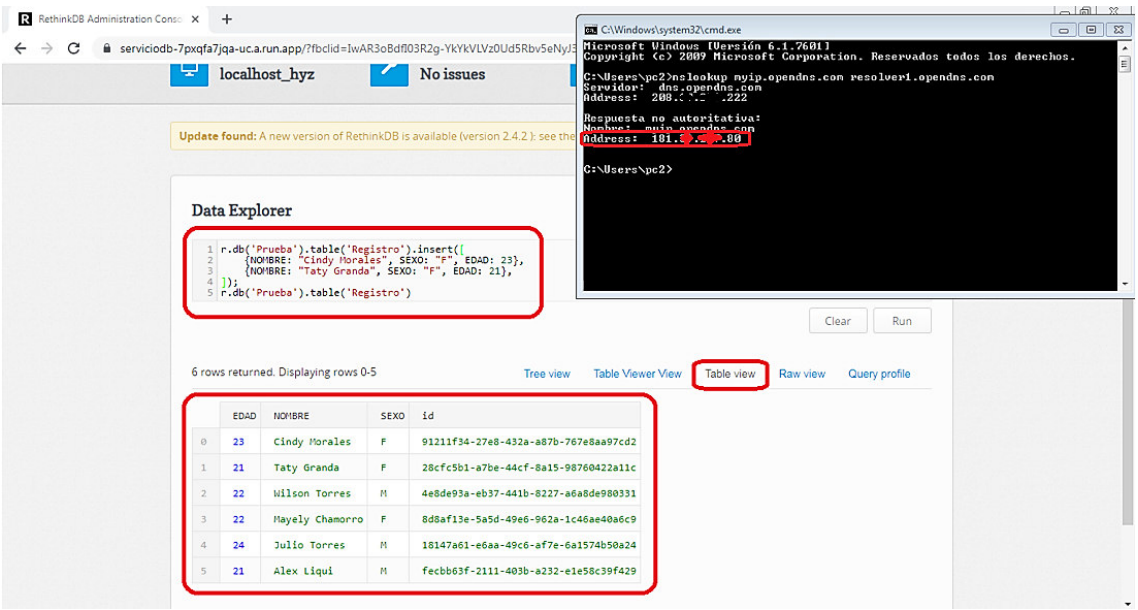

Figura 3.32 Vista de los comandos de la BD junto con la tercera red

En la Figura 3.33 dentro de la pestaña "*Tables*" en la base de datos "Prueba", dando clic al botón "*Explore*" en las estadísticas, se observa las nuevas indexaciones de registros de "Cindy Morales" y "Taty Granda" realizado en la tabla dentro de la tercera red, se verifica además el direccionamiento respectivo.

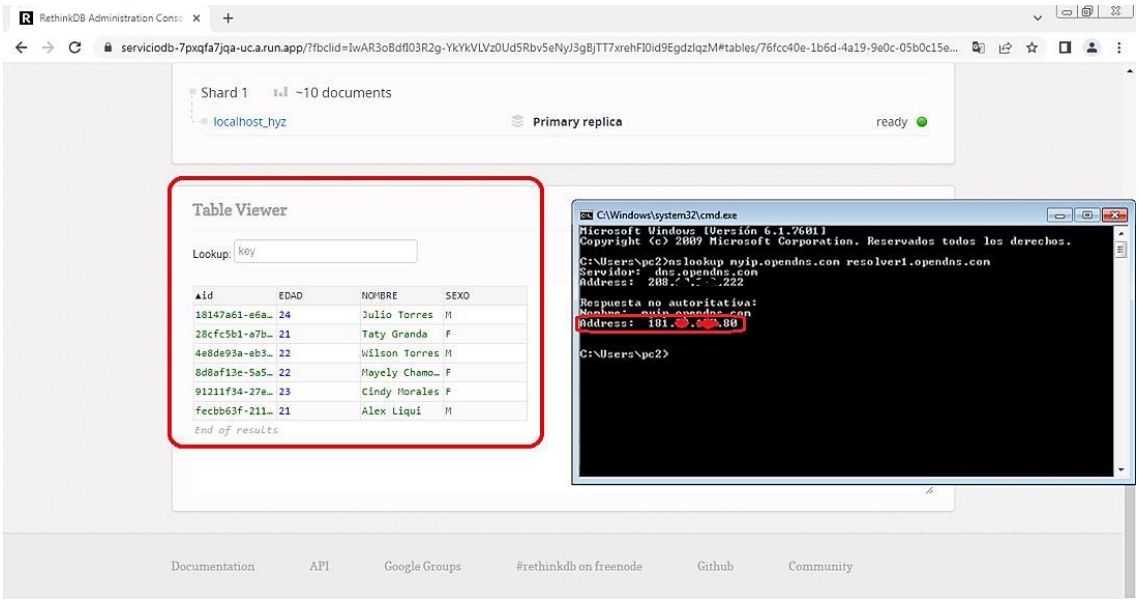

Figura 3.33 Vista de la tabla dentro de la BD junto con la tercera red

Mediante el Anexo II se observa el código QR donde se puede constatar la implementación, ejecución y las pruebas de funcionamiento de este proyecto de titulación.

4 CONCLUSIONES

- Los SGBD en la actualidad más se los conoce debido a su implementación dentro de la informática como BD sin más denotativos; son indispensables para el almacenamiento de información debido a que se puede hacer un sinnúmero de actividades dentro de ellas de forma recurrente. A la vez que se puede tener seguridad y redundancia de los datos que se recopilan en ellas, y tener así un respaldo de lo que se quiere guardar o rescatar de un negocio, actividad y/o cálculo del cual se pueda recopilar datos constantemente.
- Los contenedores han revolucionado al mundo en cuestión de la movilidad, transporte, eficacia, tamaño de las aplicaciones, así como de su implementación; junto con las plataformas que ayudan a operarlos sirven de gran ayuda para el aprendizaje y ejecución de ciertas actividades (páginas *web*, SO, BD, API, aplicaciones, etc).

- En la época actual todas las herramientas están disponibles dentro del Internet y es importante conocer las herramientas de DevOps, que pueden usarse para el manejo de contenedores y manipulación de sus especificaciones y comandos. Esto haciendo alusión a *Docker* que tiene un sinnúmero de facilidades y ventajas las cuales van desde su contenido en su página *web*, su manera de trabajar, su aplicación de escritorio y su documentación basta qué sirve para entender cosas irresolutas y/o que la comunidad ya lo ha logrado resolver. A la vez también se puede mencionar la utilidad de Kubernetes, la cual ayuda correr contenedores en forma de un archivo, donde las aplicaciones se pueden correr de manera íntegra, mediante una salida de un *link* HTTP con su respectiva IP y que logra funcionar en base de los requerimientos necesarios.
- Hablando de *Docker Hub* se puede decir que es una herramienta bastante versátil la cual tiene las imágenes con instrucciones o parámetros importantes. *Docker Hub* almacena las instrucciones válidas, comandos, puntos clave, entre otras directrices las cuales también se pueden validar de forma práctica. Hay que leer, comprender y asimilar bien estas instrucciones, debido a que si no se las entiende de una manera errónea se pueda incurrir en errores. En este contexto, se ejemplifica en la situación de que una ejecución puede salir correcta pero el usuario no podrá entender lo que pasó; o la ejecución va a ser errónea y el usuario puede pensar que está en lo correcto, por lo tanto, se debe tener presente el correcto entendimiento de las anotaciones.
- Hablando de la herramienta DevOps Kubernetes y su clúster dentro de GCP, es muy aplicable dentro de los contenedores, ya que se puede levantar dos o más funcionando al mismo tiempo. Sin embargo, se debe tomar en cuenta que estos dos contenedores van a ser independientes, y en el caso de que se deban unir se tiene que tomar en cuenta mucho la composición de la programación que va a tener el archivo o *pod* (.yml). De no ser el caso puede ser que corra el contenedor, pero uno de ellos no podrá funcionar. Para eso también existe la opción de usar *docker-compose* dentro de GCP, con sus respectivas dependencias y configuración preliminar.
- Dentro del contexto de los proveedores de la nube, se tienen una gran gama de plataformas y herramientas con diferentes características útiles para el uso e implementación de contenedores, donde la importancia radica en saber cómo usarlas y a que coste. Para este proyecto de titulación se optó por GCP debido a que fundamentalmente es una de la más populares y usadas, con una gran cantidad de información y sobre todo tiempo de uso de prueba gratuita.

- Se recalca que se usó la plantilla de RethinkDB debido a que dentro del *Cloud Run* de GCP funcionó correctamente como un enlace, haciéndolo trabajar de manera "*standalone*" (opera independientemente del sistema y/o dispositivo), debido a que esta BD funciona dentro de la *web*. Por lo tanto, su implementación y uso va de la mano con los navegadores *web* sin problemas.
- Si se requiere utilizar otras plantillas de BD dentro de GCP hay que tomar en cuenta que en caso de presentarla dentro de una página HTTPS (como lo ofrece el servicio de *Cloud Run*) la misma debe estar dirigida a ese servicio/puerto (80, 8080, 443). Debido a que ciertas plantillas pueden funcionar, pero si no son dirigidas a la manipulación en una página *web* dará un error por defecto en la implementación. A la vez ciertos SGBD funcionan mediante una interfaz *web*, sin embargo, al intentar enlazar a una BD no se va a poder debido a que GCP solo corre las imágenes, más un archivo compuesto con sus debidas dependencias.
- Para la implementación de cualquier contenedor dentro del servicio de *Cloud Run* primero debe correr el contenedor de manera interna en el CLI de GCP. Comprobado lo anterior, se verifica si funciona la implementación con su debido puerto (su número puede variar en función de la aplicación) en el *Cloud Run*. Ya implementado se debe verificar que se pueda manipular el SGBD de manera correcta. Debido a que puede funcionar la implementación o enlace, pero al momento de ejecutarla va a dar muchos errores de conexión o no puede funcionar la indexación de registros.
- Dentro del servicio de *Cloud Run* de GCP para la implementación de estos contenedores existen dos errores comunes que pueden salir. El primero es el del puerto y el segundo es con respecto a la programación o la ejecución dentro de GCP, donde la misma plataforma tiene información sobre la resolución de los mismos. El error de puerto de las BD relacionales SQL que tienen el puerto 3306 no se logran implementar dentro de GCP. Mientras que el segundo error de ejecución tiene que ver con un error dentro del contenido de la imagen del contenedor y se tendría que optar por otra imagen.
- Dentro de las verificaciones se pudo constatar que dentro de la red del autor la base de datos de RethinkDB (mediante un *link*) funcionó de manera idónea, ya que se creó una BD, una tabla y registros en la misma. A la vez que se pudo indexar nuevos datos/registros en la tabla en otras dos redes diferentes, mediante la utilización del enlace resultante de GCP. Con esto se verificó que el presente proyecto de titulación funciona de manera correcta.

5 RECOMENDACIONES

- Se recomienda tomar en cuenta de una forma general, que no solo existe la plataforma de GCP para la implementación de contenedores actualmente. Existen plataformas con diferentes características las cuales se pueden ir probando en función de lo que se requiere, por afinidad o familiarización.
- Para poder trabajar dentro del Kubernetes en su clúster dentro de GCP, existe bastante información a través de páginas y tutoriales para poder programar las distintas funciones y características, que se requieran dentro de los contenedores a implementar. Donde lo importante es saber los requerimientos, en función de los servicios o herramientas que se vayan a necesitar, además del concomimiento de los comandos, estructuras, sintaxis y ubicación de los mismos para el correcto desenvolvimiento.
- También se puede sugerir que, dentro del contexto de los SGBD de manera general, se debe localizar la información dentro de páginas oficiales y confiables. RethinkDB a pesar de que es una BD nueva posee muchas ventajas; es conveniente buscar dentro de varias fuentes como videos, documentos, presentaciones u otras páginas relacionadas a RethinkDB para tener conocimiento más detallado sobre la misma.
- Se aconseja que para la utilización de toda la información de *Docker* se debe tener en claro qué es lo que se va a realizar específicamente, debido a que cuenta con mucha información. A su vez de que se debe entender todos los parámetros con respecto a sus comandos y la función que conlleva cada uno, así como su estructura, sintaxis y ubicación.
- Para poder trabajar dentro de GCP de manera óptima se debe tener una esquematización de lo que se va hacer, en función del proyecto en donde se encuentra. Debido a que si bien tiene un período de prueba gratuito largo se debe aprovechar al máximo sus capacidades, donde en caso de estar estancado en algún procedimiento estándar, la misma plataforma ofrece soluciones diferentes para solventarlo con el conocimiento previo de cómo usarlo.

6 REFERENCIAS BIBLIOGRÁFICAS

[1] NetApp, «NetApp,» [En línea]. Available: https://www.netapp.com/es/devops-solutions/what-are-containers/#:~:text=Los%20contenedores%20son%20una%20forma,una%20aplicaci%C3%B3n%20de%20mayor%20tama%C3%B1o.. [Último acceso: 18 Abril 2022].

[2] Azure, «Azure,» [En línea]. Available: https://azure.microsoft.com/es-es/overview/what-is-a-container/#overview. [Último acceso: 18 Abril 2022].

[3] IBM Cloud Education, «IBM,» 23 Junio 2021. [En línea]. Available: https://www.ibm.com/cl-es/cloud/learn/containers. [Último acceso: 18 Abril 2022].

[4] Red Hat, «Red Hat,» 15 Enero 2020. [En línea]. Available: https://www.redhat.com/es/topics/containers/containers-vs-vms#:~:text=elegir%20Red%20Hat%3F-,Resumen,contenedores%20se%20miden%20en%20megabytes.. [Último acceso: 18 Abril 2022].

[5] K. A. P. Chicaiza, «IMPLEMENTACIÓN DE SERVIDORES DNS Y WEB BASADO EN CONTENEDORES ALOJADOS EN LA NUBE,» Quito, 2022.

[6] R. Rivera, *Clases de Intranets,* 2021.

[7] Red Hat, «Red Hat,» [En línea]. Available: https://www.redhat.com/es/topics/cloud-native-apps. [Último acceso: 18 Junio 2022].

[8] Universidad UNADE, «UNADE,» 24 Septiembre 2019. [En línea]. Available: https://unade.edu.mx/que-es-la-gestion-de-base-de-datos/. [Último acceso: 14 Mayo 2022].

[9] R. Marín, «Revistadigital IINESEM BUSINESS SCHOOL,» 16 Abril 2019. [En línea]. Available: https://www.inesem.es/revistadigital/informatica-y-tics/los-gestores-de-bases-de-datos-mas-usados/. [Último acceso: 18 Abril 2022].

[10] J. Jordana, «openexpo | europe,» [En línea]. Available: https://openexpoeurope.com/es/rethinkdb-bbdd-para-web-tiempo-real/. [Último acceso: 18 Junio 2022].

[11] Redigit, «Redigit,» [En línea]. Available: https://blog.redigit.es/tecnologias-de-virtualizacion-basada-en-contenedores/. [Último acceso: 29 Mayo 2022].

[12] Kenia, «Instinto Programador,» 16 Diciembre 2020. [En línea]. Available: https://www.instintoprogramador.com.mx/2020/12/5-excelentes-soluciones-de-bases-de.html. [Último acceso: 18 Junio 2022].

[13] GeoBolivia, «GeoBolivia,» [En línea]. Available: http://geo.gob.bo/portal/?La-importancia-de-utilizar-Docker-en-las-aplicaciones-web-299. [Último acceso: 29 Mayo 2022].

[14] atareao, «ATAREAO,» 10 Agosto 2021. [En línea]. Available: https://atareao.es/tutorial/podman/#:~:text=Podman%20es%20una%20herramienta%20nativa,e%20im%C3%A1genes%20Open%20Containers%20Initiative%20(. [Último acceso: 18 Julio 2022].

[15] L. Cano, «PANDORAFMS,» 15 Marzo 2022. [En línea]. Available: https://pandorafms.com/blog/what-is-podman/. [Último acceso: 18 Julio 2022].

[16] Disaster Project, «Disaster Project,» 03 Julio 2019. [En línea]. Available: https://www.disasterproject.com/docker-o-podman/. [Último acceso: 18 Julio 2022].

[17] Isaac, «Desde Linux,» [En línea]. Available: https://blog.desdelinux.net/docker-vs-kubernetes/#:~:text=Kubernetes%20ofrece%20mejor%20flexibilidad%2C%20incluso,es%20complicado%2C%20Docker%20m%C3%A1s%20sencillo.. [Último acceso: 29 Mayo 2022].

[18] Salesforce, «salesforce,» [En línea]. Available: https://www.salesforce.com/mx/cloud-computing/. [Último acceso: 03 Junio 2022].

[19] T. Grapsas, «rockcontent,» 16 Septiembre 2018. [En línea]. Available: https://rockcontent.com/es/blog/computacion-en-la-nube/. [Último acceso: 03 Junio 2022].

[20] V. Trafaniuc, «Maplink,» 21 Diciembre 2021. [En línea]. Available: https://maplink.global/blog/es/que-es-google-cloud/. [Último acceso: 03 Junio 2022].

[21] Google Cloud, «Google Cloud,» [En línea]. Available: https://cloud.google.com/. [Último acceso: 03 Junio 2022].

[22] M. Gimenez, «hiberusblog,» 20 Julio 2020. [En línea]. Available: https://www.hiberus.com/crecemos-contigo/amazon-web-services-aws-que-es-y-que-ofrece/. [Último acceso: 03 Junio 2022].

[23] AWS, «AWS,» [En línea]. Available: https://aws.amazon.com/es/free/start-your-free-trial/#:~:text=La%20capa%20gratuita%20de%20AWS,costo%20alguno%20durante%2012%20meses.. [Último acceso: 03 Junio 2022].

[24] conzultek, «conzultek,» [En línea]. Available: https://blog.conzultek.com/microsoft-azure-que-es-como-funciona-como-ayuda-a-las-empresas. [Último acceso: 03 Junio 2022].

[25] Docker, «Docker Hub,» [En línea]. Available: https://hub.docker.com/search?q=data%20bases&page=1. [Último acceso: 12 Mayo 2022].

[26] Docker, «Docker Hub,» [En línea]. Available: https://hub.docker.com/_/rethinkdb. [Último acceso: 12 Junio 2022].

[27] «stack overflow,» [En línea]. Available: https://stackoverflow.com/questions/43697810/meaning-of-pwd-path-to-directory. [Último acceso: 27 Junio 2022].

[28] bradtraversy, «Github Gist,» [En línea]. Available: https://gist.github.com/bradtraversy/6aef077f93c48465891f12958b84a22d. [Último acceso: 30 Junio 2022].

7 ANEXOS

ANEXO I: Certificado de Originalidad

CERTIFICADO DE ORIGINALIDAD

Quito, D.M. 25 de agosto de 2022

De mi consideración:

Yo, GABRIELA KATHERINE CEVALLOS SALAZAR, en calidad de Director del Trabajo de Integración Curricular titulado IMPLEMENTACIÓN DE UNA BASE DE DATOS MEDIANTE CONTENEDORES ALOJADOS EN LA NUBE elaborado por el estudiante WILSON DAVID TORRES CAÑAR de la carrera en TECNÓLOGÍA SUPERIOR EN REDES Y TELECOMUNICACIONES, certifico que he empleado la herramienta Turnitin para la revisión de originalidad del documento escrito completo, producto del Trabajo de Integración Curricular indicado.

El documento escrito tiene un índice de similitud del 11%.

Es todo cuanto puedo certificar en honor a la verdad, pudiendo el interesado hacer uso del presente documento para los trámites de titulación.

NOTA: Se adjunta el link del informe generado por la herramienta Turnitin.

https://epnecuador-my.sharepoint.com/:f:/g/personal/gabriela_cevalloss_epn_edu_ec/EsucPuJ4VklKuANeMQr3Fd4Bv9hM4XypLONxHVvcbdGC0w?e=OTusaD

Atentamente,

Gabriela Katherine Cevallos Salazar

Docente

Escuela de Formación de Tecnólogos

ANEXO II: Enlaces

Anexo II.I Código QR de la implementación y pruebas de funcionamiento

ANEXO III: Códigos Fuente

Anexo III.II Líneas de comandos para la creación de la base de datos ('Prueba') y una tabla ('Registro')

```
r.dbCreate('Prueba');

r.db('Prueba').tableCreate('Registro');
```

Anexo III.IIII Líneas de comandos para la indexación de campos en la tabla ('Registro')

```
r.db('Prueba').table('Registro').insert([
    {NOMBRE: "Wilson Torres", SEXO: "M", EDAD: 22},
    {NOMBRE: "Julio Torres", SEXO: "M", EDAD: 24},
    {NOMBRE: "Alex Liqui", SEXO: "M", EDAD: 21},
]);
```