Profesor:

Jesse Padilla Agudelo

Integrantes:

Brenda Catalina Barahona Pinilla (201812721)
Juan Diego González Gomez (201911031)
Kevin Steven Gamez Abril (201912514)
Sergio Julian Zona Moreno (201914936)

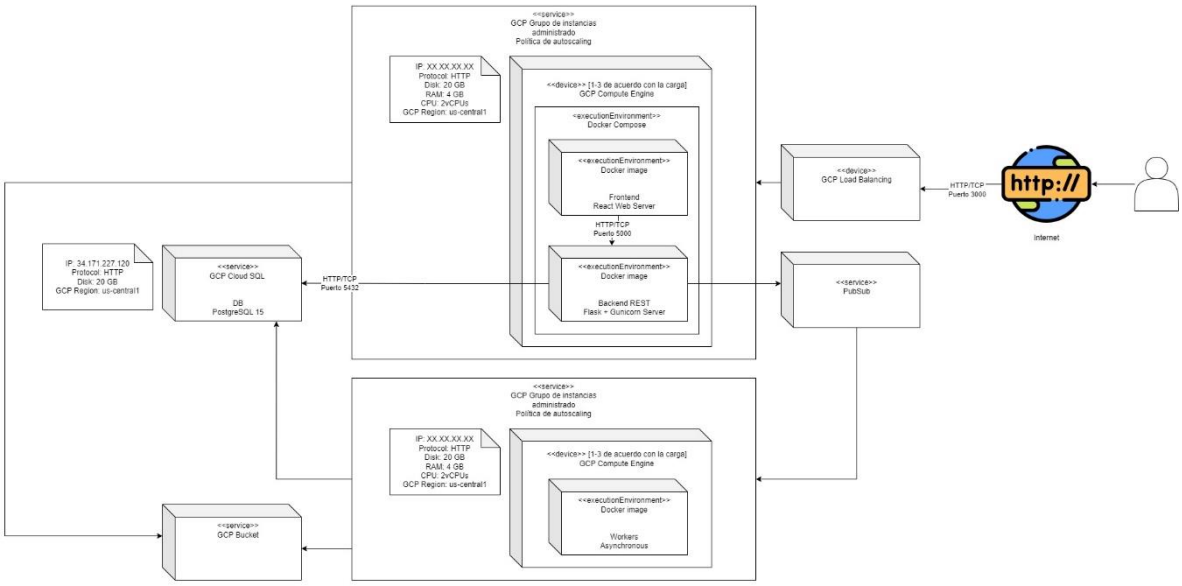
Tabla de contenido

1 Introducción

En este documento se documenta la solución a la Entrega 3 del proyecto del curso *Desarrollo de soluciones cloud*. El propósito central es desplegar una aplicación Web completa (Frontend + Backend + Base de datos relacional + Pub/Sub + Cloud Storage + Procesamiento asíncrono) utilizando servicios de GCP.

2 Arquitectura de Software

2.1 Diagrama de despliegue y descripción



La arquitectura de Software que se adoptó fue la misma que recomendó el profesor Jesse en clase. Para simplificar la arquitectura la dividiremos en los diferentes servicios:

- Grupos de autoscaling: se crearon dos grupos de autoscaling para esta entrega. El primero corresponde al grupo de Autoscaling que maneja las instancias del Front y el Back. El otro grupo corresponde al conjunto de instancias que maneja la capa asincrónico de los Workers. Ambos grupos se conectan por medio del PubSub. Más adelante daremos detalles del balanceo de carga y escalamiento.
- Cloud Storage Bucket: funciona como sistema de almacenamiento unificado al cual se conecta el backend de la aplicación para extraer rápidamente los archivos, y el servicio de procesamiento de archivos el cual almacena los PDFs una vez fueron procesados.
- Servicio de procesamiento de archivos (asincrónico): con el fin de mantener la tarea de procesamiento de archivos de manera separada del servidor principal, se delegó esta tarea a una grupo de máquinas virtuales. La conexión se realiza a través de Pub/Sub, donde el backend envía un mensaje al tópico y la capa de procesamiento asíncrono se suscribe al tópico donde es notificado de los archivos que debe procesar.
- Base de datos relacional: utilizada en previas entregas, utilizamos como servicio GCP Cloud SQL, específicamente una base de datos con PostgreSQL 15.
- Balanceador de carga: recibe las peticiones directamente del usuario y las redirige al conjunto de instancias con la política de autoscaling.

Con estos servicios se da cumplimiento a los componentes que se solicitaban en el taller.

2.2 Características de las máquinas y las instancias

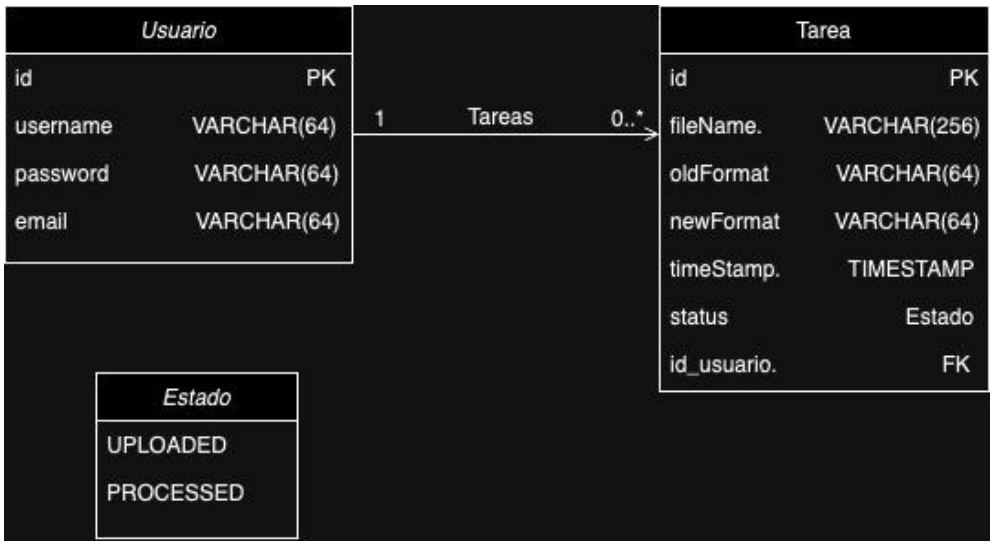
Todas las máquinas tienen las siguientes características:

- SO: Ubuntu 22.04 LTS
- Protocolo de conexión: HTTP
- Disk: 10 GB
- RAM: 614 MB
- CPU: 1vCPUs
- GCP Region: us-central1

2.3 Balanceo de carga y escalamiento

Implementamos una política de autoescalado en para gestionar eficientemente la carga de nuestra aplicación. Cuando el uso del procesador supera el 60%, se despliega automáticamente otra máquina virtual basada en una plantilla predefinida. Del mismo modo, los workers de procesamiento de archivos se escalan dinámicamente de 1 a un máximo de 3 máquinas según la demanda. Además implementamos un balanceo de carga al inicio del flujo de tráfico. Este distribuye las solicitudes de los usuarios entre las maquinas disponibles en ese momento. Cuando una nueva máquina virtual se despliega automáticamente debido a la alta carga de procesamiento, el balanceo de carga redirige el tráfico hacia ella, asegurando una distribución uniforme de la carga y evitando la sobrecarga de cualquier instancia específica.

2.4 Diagrama entidad relación



El modelo presentado en la imagen describe un sistema de gestión de usuarios y tareas, estructurado en dos tablas principales: Usuario y Tarea, con una relación de uno a muchos. Esto significa que un usuario puede tener asignadas múltiples tareas, pero cada tarea está específicamente vinculada a un solo usuario. La tabla Usuario incluye campos para el identificador único del usuario, nombre de usuario, contraseña y correo electrónico, siendo este último único para cada usuario. Por su parte, la tabla Tarea consta de un identificador

único, el nombre del archivo asociado, los formatos original y nuevo del archivo, una marca de tiempo que registra la creación o modificación de la tarea, el estado de la tarea que puede ser 'UPLOADED' (subido) o 'PROCESSED' (procesado), y una clave foránea que enlaza cada tarea a su usuario correspondiente.

2.4 Release

https://github.com/JuanDiegoGonzalez/Proyecto_Cloud_Grupo2/releases/tag/3.0

2.5 Video de sustentación

https://www.youtube.com/watch?v=cqIQU9DGk0A

3 Pruebas de estrés, análisis y documentación

Para efectos prácticos seleccionamos JMeter como herramienta de pruebas que nos permitirá evaluar los dos escenarios planteados en la entrega previa.

Escenario	Descripción	Subescenario	Tipo	Header	Número de peticiones	Latencia máxima	Error máximo	Servicio(s) que afecta
Escenario 1	Peticiones POST para subir un archivo	Escenario 1.1	POST	JSON	1000	300 ms	1%	VM 1, VM 2, Cloud SQL y Linux NFS
		Escenario 1.2	POST	JSON	2000	500 ms	2%	VM 1, VM 2, Cloud SQL y Linux NFS
Escenario 2	Peticiones GET para descargar un archivo	Escenario 2.1	GET	JSON	1000	300 ms	1%	VM 1, Cloud SQL y Linux NFS
		Escenario 2.2	GET	JSON	2000	500 ms	2%	VM 1,Cloud SQL y Linux NFS

Conclusiones del escenario 2:

Para este escenario en particular pudimos observar que el % error esperado se mantuvo en 0% hasta las 8000-8500 peticiones (siendo este el punto de quiebre). Lo cuál indica que contamos con una capacidad mayor para atender usuarios simultáneamente respecto a la entrega 1. Los tiempos de latencia promedio aumentar un poco, pero se mantuvieron en el umbral esperado.

Se planteo de SLO de menos del 5% de error en alta demanda. Nuestro SLI para 8000 peticiones fue de 3.83% por lo que cumplimos de manera satisfactoria el SLO.

3.1 Escenario 1

Conclusiones del escenario 1:

A diferencia de la entrega pasada, el escenario 1 mejoró sus métricas de desempeño como esperábamos. Encontramos una menor latencia y un menor porcentaje de error. Asimismo la funcionalidad logra aceptar una mayor cantidad de *threads* de ejecución simultánea.

Conclusiones del escenario 2:

Como era de esperarse, el escenario 1 obtuvo métricas inferiores en términos de desempeño de la aplicación. Esto se debe a que afectaba más servicios, por lo que la latencia aumentó en más de un 20%. Asimismo el % error logra satisfacer la alta demanda con más de 8000 a diferencia de la entrega pasada.

4 Limitaciones y consideraciones adicionales

4.1 Limitaciones

Dentro de las limitaciones podemos encontrar que:

- Las pruebas de carga se ejecutan con el software de JMeter por lo que nos encontramos limitados a las funcionalidades que este disponga.
- Según la ubicación de los servidores y la disponibilidad que tengamos, puede que la latencia aumente debido a la distancia que se tiene con la zona de disponibilidad.
- La capacidad de las máquinas es limitada, debido a el número de créditos disponibles. De igual forma se tiene en cuenta que esta aplicación es de tamaño pequeño y para fines académicos. Asimismo, tampoco podemos aprovisionar múltiples máquinas por lo que no es posible realizar pruebas de escalabilidad (y autoescalabilidad) y disponibilidad dentro del alcance del curso.
- Los grupos de autoscaling tampoco son lo suficientemente potentes. Adicionalmente, existe una limitación en el tiempo que dura el grupo en activar la funcionalidad de autoscaling.

4.2 Consideraciones adicionales para escalar la aplicación

- Ajustar (de acuerdo con los rubros disponibles para el curso) la capacidad de las máquinas especialmente en temas de procesamiento y memoria RAM, es decir, realizar un escalamiento vertical de las instancias.
- Modificar la política de *autoscaling* que genere escalamiento tanto horizontal como vertical cuando detecte picos de peticiones superiores a un umbral del 85% de capacidad. Esta política de *autoscaling* se puede complementar con modelos de inteligencia artificial que permitan detectar de manera eficiente los picos de peticiones.