Entrega 5 - Arquitectura, conclusiones y consideraciones

Javier Alejandro Gómez, Alejandra Guerrero, Diego Alejandro Peña, Juan Ignacio Arbeláez

{ja.gomez1003, a.guerrero10, da.pena20, ji.arbelaez}@uniandes.edu.co

Desarrollo de soluciones cloud - MINE

Universidad de los Andes, Bogotá, Colombia

Fecha de presentación: mayo 28 de 2023

Tabla de contenido

1. Introducción

En el presente documento se expone la arquitectura implementada para generar una aplicación web en Flask, la cual permita solicitar y realizar la compresión de archivos por medio de peticiones HTTP y utilizando un enfoque de procesamiento asíncrono. También se presentan las consideraciones que se tuvieron en cuenta en el desarrollo de este proyecto para cumplir con el objetivo, el cual era aplicar los conceptos de escalabilidad, una de las características principales de los ambientes en nube.

El código y la documentación de este proyecto se pueden consultar en los siguientes enlaces:

- Repositorio GitHub: https://github.com/aguerrero10/Desarrollo-cloud-grupal/tree/entrega_final
- Documentación de Postman: https://documenter.getpostman.com/view/13843294/2s93CPrY2y
- Video de sustentación: Link del video

2. Arquitectura planteada

En este trabajo se ha desarrollado una arquitectura robusta y escalable para la gestión de datos masivos utilizando los siguientes componentes:

- Base de datos PostgreSQL usando el servicio Cloud SQL de GCP
- Para el almacenamiento de archivos se usó Cloud Storage
- Para el despliegue de los workers se está usando Cloud Run
- Para el despliegue de la aplicación web se está usando Cloud Run
- Se usa Pub/Sub para la comunicación entre la aplicación web y el worker
- Por último, Cloud Monitoring para el monitoreo del proyecto y sus componentes

En el siguiente diagrama se puede evidenciar la arquitectura de la aplicación:

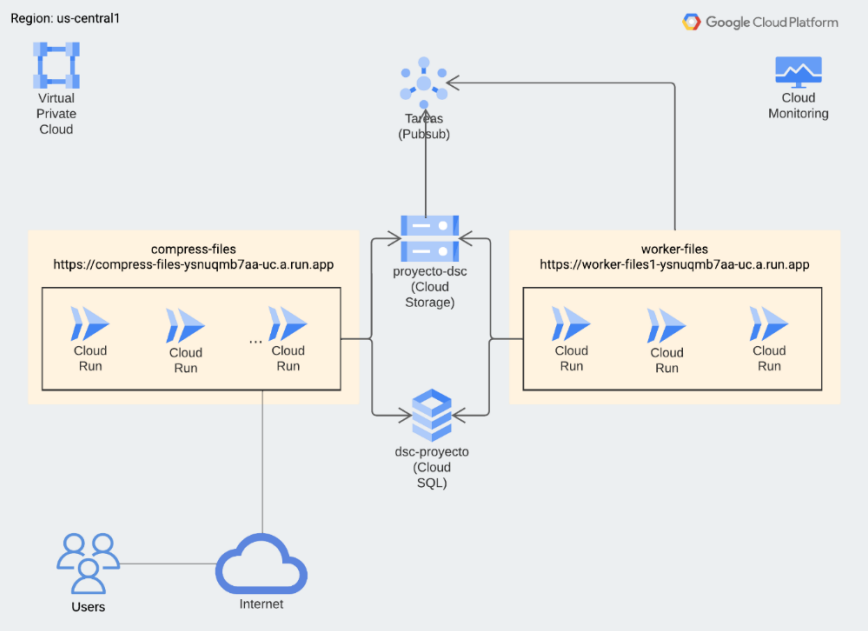


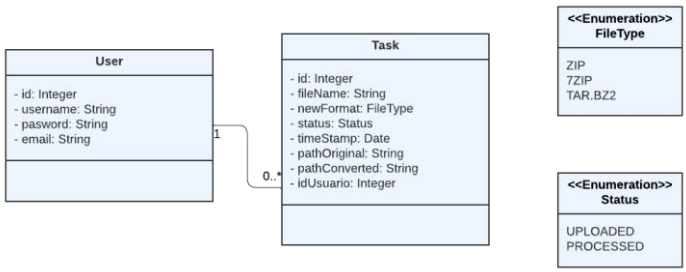
Figura 1. Diagrama de despliegue

Como se evidencia en el diagrama de despliegue, todos los componentes pertenecen a una misma región, y se interconectan entre sí usando una red privada VPC. La capa web server de la aplicación está desplegada en Cloud Run y responde a las solicitudes HTTP para la creación de usuarios y la gestión de compresión de archivos. El servicio de Cloud Run en donde está desplegada la capa web se conecta con Cloud SQL para poder almacenar de forma independiente y transversal a la capa de procesamiento los metadatos de los usuarios y de sus solicitudes de compresión, así como también se conecta con Cloud Storage para la persistencia de los archivos pendientes de comprimir y comprimidos.

Por otra parte, la capa worker también fue desplegada en Cloud Run, este servicio realiza la compresión de los archivos que fueron solicitados por medio de la aplicación web y se activa gracias a la suscripción de un tema de Pub/Sub, el cual recibe una notificación cuando se carga un archivo en Cloud Storage. Es entonces cuando el worker se encarga de comprimir el archivo, actualizar el estado de esa tarea en la base de datos y notificar por correo electrónico al usuario para que proceda a descargar su archivo en versión comprimida.

Por otro lado, la lógica de negocio (backend) y la base de datos siguen el modelo de datos que se muestra en el siguiente diagrama:

Diagrama de clases



Es decir, el modelo de datos se compone de la entidad Usuario la cual se compone de sus datos básicos y puede o no tener múltiples tareas de compresión de archivos. Por otra parte, la entidad tarea tiene los datos del archivo a comprimir, su estado y tipo de archivo de archivo de compresión, modeladas con una enumeración.

3. Consideraciones de la aplicación

Se configuró y desplego la aplicación web en Google Cloud Run. Para lograrlo, se creó el documento Dockerfile y se desplego con la función "`gcloud run deploy`" (ver anexo 1). La descripción de la configuración final es:

```
✓ Service compress-files in region us-central1
URL:     https://compress-files-ysnuqmb7aa-uc.a.run.app
Ingress: all
Traffic:
  100% LATEST (currently compress-files-00005-wed)
 Last updated on 2023-05-28T02:22:21.705523Z by a.guerrero10@uniandes.edu.co:
  Revision compress-files-00005-wed
  Image:             gcr.io/dsc-proyecto/compress-files
  Port:              5000
  Memory:            512Mi
  CPU:               1000m
  Service account:   357876382465-compute@developer.gserviceaccount.com
  Concurrency:       80
  Max Instances:     3
  Timeout:           300s
  VPC connector:
    Name:          projects/dsc-proyecto/locations/us-central1/connectors/conector-acceso-sql
    Egress:          private-ranges-only
  Startup Probe:
    TCP every 240s
    Port:            5000
    Initial delay:   0s
    Timeout:         240s
    Failure threshold: 1
```

Para configurar y desplegar el worker en Google Cloud Run, se creó el tema 'tareas' y se creó la subscripción 'myWorkerSubs'. Se implementó un controlador con flask que procesa los mensajes de Pub/Sub. Luego, se configuró un container usando un Dockerfile y se desplego con la función "`gcloud run deploy`" (ver anexo 1). La descripción de la configuración final es:

```
✓ Service worker-files1 in region us-central1
URL:     https://worker-files1-ysnuqmb7aa-uc.a.run.app
Ingress: all
Traffic:
  100% LATEST (currently worker-files1-00013-mob)

Last updated on 2023-05-28T22:19:17.733892Z by a.guerrero10@uniandes.edu.co:
  Revision worker-files1-00013-mob
  Image:             gcr.io/dsc-proyecto/worker-files1
  Port:              8080
  Memory:            512Mi
  CPU:               1000m
  Service account:   357876382465-compute@developer.gserviceaccount.com
  Concurrency:       80
  Max Instances:     3
  Timeout:           300s
  VPC connector:
    Name:          projects/dsc-proyecto/locations/us-central1/connectors/conector-acceso-sql
    Egress:          private-ranges-only
  Startup Probe:
    TCP every 240s
    Port:            8080
    Initial delay:   0s
    Timeout:         240s
    Failure threshold: 1
```

Para que la aplicación anterior pueda escalar a cientos de usuarios finales es importante:

- Separar la capa de almacenamiento de la capa de procesamiento. En este caso, se utilizó Cloud Storage y Cloud SQL para la persistencia y centralización de los datos estructurados y no

estructurados. De esta forma, se pueden utilizar diversas herramientas para implementar la lógica del negocio (anteriormente se usó Compute Engine y Cloud Functions, y en este escenario se usó Cloud Run), y evitar redundancia e inconsistencia en los datos, así como crear o eliminar instancias de procesamiento de acuerdo con el nivel de uso del servicio.

- Definir la estrategia de auto escalamiento. Esto dependerá del servicio a utilizar, en este caso, al usar Cloud Run no fue necesario definir temas de infraestructura hardware, ni configurar balanceadores de carga basados en el uso de CPU, número de solicitudes, etc. No obstante, aunque Cloud Run determinó en qué momento es más eficiente escalar hacia arriba o hacia abajo, es importante conocer el nivel de uso real de la aplicación para limitar el número mínimo y máximo de instancias que Cloud Run puede desplegar.
- Límites de ejecución. Así como se analiza el escenario de éxito, los escenarios de error también deben tener un camino de ejecución para no generar un efecto domino en la disponibilidad de los servicios. Por ejemplo, al configurar la activación de Cloud Run a partir de la suscripción a Pub Sub es importante definir el intervalo de tiempo entre intentos de ejecución o el número máximo de intentos que se realizará en caso de fallo para que no se genere un ciclo infinito de tareas que no se pueden procesar y aumentar drásticamente las instancias activas, y en consecuencia, los costos del proyecto.
- Conexión entre servicios y permisos. Aspectos de lógica de negocio se pueden evaluar fácilmente en un computador local o arquitectura de monolítico. Sin embargo, el uso de múltiples servicios agrega el reto de tener presente la forma en que se van a conectar, es decir, establecer redes y subredes privadas que minimicen la latencia, definir el protocolo y puerto más acorde, ya que los servicios de GCP no suelen interconectarse de la misma forma; finalmente, crear usuarios o cuentas de servicio con sus permisos suficientes para la conexión y ejecución segura y eficaz de los componentes. Estos elementos se deben automatizar y eliminar la dependencia de un humano para que la escalabilidad sea posible.
- Uso de herramientas de terceros. Utilizar funcionalidades externas, por ejemplo, correo electrónico, agrega ciertas restricciones como la cantidad máxima de correos que se puede enviar, el tamaño máximo del mensaje o de los archivos que se pueden adjuntar, entre otros. Estas restricciones afectan el rendimiento y éxito de los servicios propios, especialmente en contextos de alta escalabilidad.

4. Escenarios de pruebas

Prerrequisitos:

- Aplicación de compresión de archivos funcionando en GCP
- Proyecto para pruebas de carga en Jmeter
- Archivos con información de Usuarios y Tareas con el que se enviarán las solicitudes

username	password	email
user1	pass1	dummy1@gmail.com
user2	pass2	dummy2@gmail.com
user3	pass3	dummy3@gmail.com
user4	pass4	dummy4@gmail.com

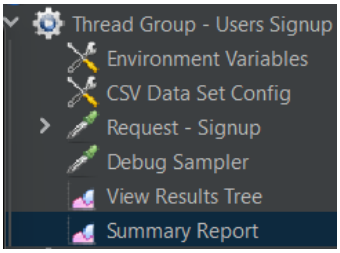
username	password	format	file
user1	pass1	ZIP	TestFile1.txt
user2	pass2	SEVENZIP	TestFile2.txt
user3	pass3	ZIP	TestFile3.txt
user4	pass4	SEVENZIP	TestFile4.txt

4.1. Escenario 1 – Pruebas de Escalabilidad Parte A, es decir, pruebas de carga solo con los ajustes de escalabilidad a la capa web.

Descripción

Con el fin de realizar una carga web a la aplicación, anteriormente se definió un escenario específico para el registro de usuarios mediante el uso de un archivo CSV que contenía la información requerida

para el registro de los usuarios en la aplicación. Esto genera una carga a la parte web específicamente a la funcionalidad de registro de usuarios.



Configuración usada

Con el fin de observar un comportamiento similar a las pruebas de carga previamente realizadas, se configuró la herramienta con los siguientes parámetros:

- Número de usuarios concurrentes: 12
- Número de solicitudes totales enviadas: 960

Resultados

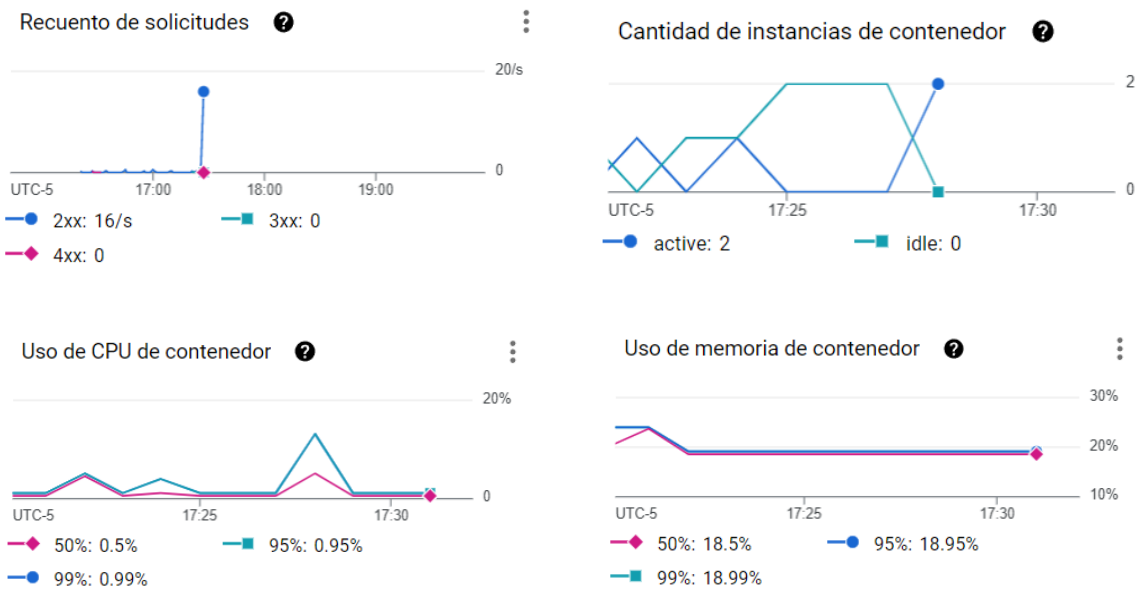
Al enviar la carga con la configuración mencionada, se observó que el sistema procesó todas las solicitudes enviadas con una carga menor del 20% de uso para el componente "Cloud Run" en donde se procesa la capa web. Por otra parte, la base de datos tuvo un gran uso de lectura. Las métricas obtenidas en Jmeter tras el procesamiento para el número de solicitudes descrito fueron:

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/s...	Sent KB/sec	Avg. Bytes
Request - Sign...	960	118	97	270	20.92	0.00%	68.0/sec	41.90	27.20	630.9
Debug Sampl...	960	0	0	1	0.20	0.00%	69.0/sec	48.94	0.00	726.4
TOTAL	1920	59	0	270	61.19	0.00%	136.0/sec	90.14	27.20	678.7

A continuación, se puede observar el pico de procesamiento de los diferentes componentes.

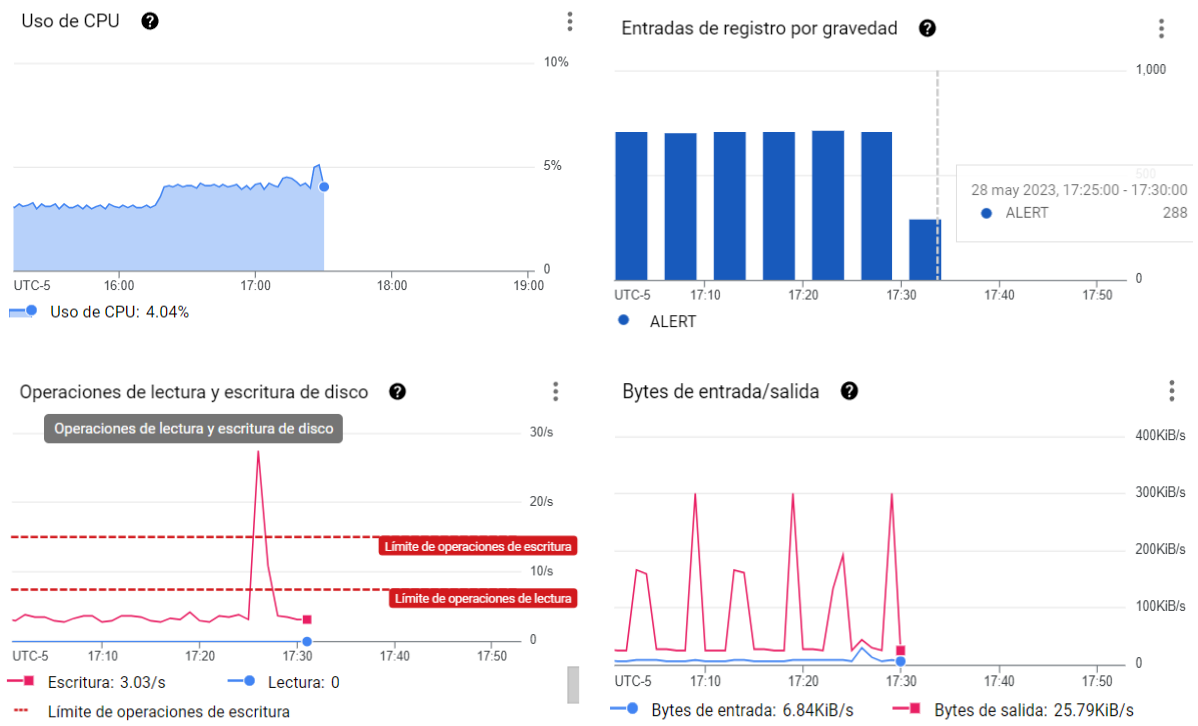
Cloud Run – Web server

A continuación, se puede observar que el uso de CPU de este componente fue menor al 20% como se mencionó anteriormente, lo que podría permitir el procesamiento de una cantidad mucho mayor en la capa web. Asimismo, el uso de Memoria fue de 20% aproximadamente.



Cloud SQL

La base de datos tuvo un pico de lectura mucho mayor al límite definido debido a que toda la información de usuarios se almacena en esta base de datos. El uso de la CPU igual o menor al 5%, por lo que se dice que el componente fue capaz de procesar todas las solicitudes.



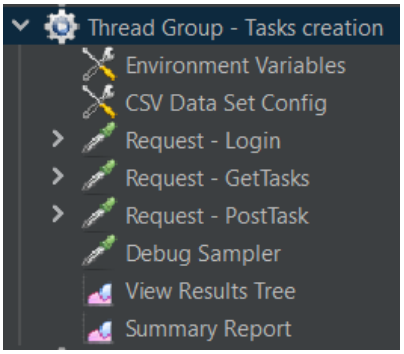
Finalmente, se puede observar en la siguiente imagen una parte de la tabla en donde se crearon los usuarios tras la prueba de carga:

id [PK] integer	username character varying	email character varying	password character varying
51	user3	dummy3@gmail.com	pass3
52	user4	dummy4@gmail.com	pass4
53	user5	dummy5@gmail.com	pass5
54	user6	dummy6@gmail.com	pass6
55	user7	dummy7@gmail.com	pass7
56	user8	dummy8@gmail.com	pass8
57	user9	dummy9@gmail.com	pass9
58	user10	dummy10@gmail.com	pass10

4.2. Escenario 2 – Pruebas de Escalabilidad Parte B, es decir, pruebas de carga a la aplicación considerando la escalabilidad de la capa web y la capa de procesamiento asíncrono.

Descripción

Con el fin de realizar una carga tanto en la aplicación web como en el procesamiento asíncrono de los datos, anteriormente se definió un escenario específico para el inicio de sesión y envío de archivos por parte de los usuarios. Mediante el uso de un archivo CSV se especificó la información requerida para el inicio de sesión de los usuarios en la aplicación, así como el formato de compresión deseado y el archivo específico. A continuación, se puede ver una imagen del procesamiento creado en la herramienta Jmeter para el procesamiento de este escenario:



Configuración usada

Con el fin de observar un comportamiento similar a las pruebas de carga previamente realizadas, se configuró la herramienta con los siguientes parámetros:

- Número de usuarios concurrentes: 12
- Número de solicitudes totales enviadas: 960
- Tamaño del archivo: 5MB

Resultados

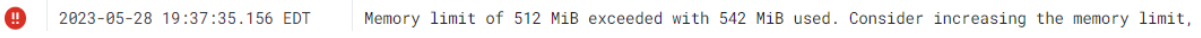
No todas las solicitudes masivas para la subida del archivo fueron procesadas, obteniendo un error del 1.04%. Esto puede deberse a limitaciones de hardware para el componente de "Cloud Run – Worker" definidas durante la creación de este. A continuación, se puede observar las métricas obtenidas:

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/s...	Sent KB/sec	Avg. Bytes
Request - Log...	960	138	86	3962	198.01	0.00%	3.8/sec	2.36	1.28	628.0
Request - Get...	960	145	90	4466	200.47	0.00%	3.9/sec	1.07	2.08	281.0
Request - Post...	960	2080	586	61170	6533.84	1.04%	3.9/sec	2.02	19487.61	531.7
Debug Sampl...	950	0	0	1	0.23	0.00%	3.9/sec	3.01	0.00	795.0
TOTAL	3830	592	0	61170	3385.89	0.26%	15.3/sec	8.34	19173.37	558.3

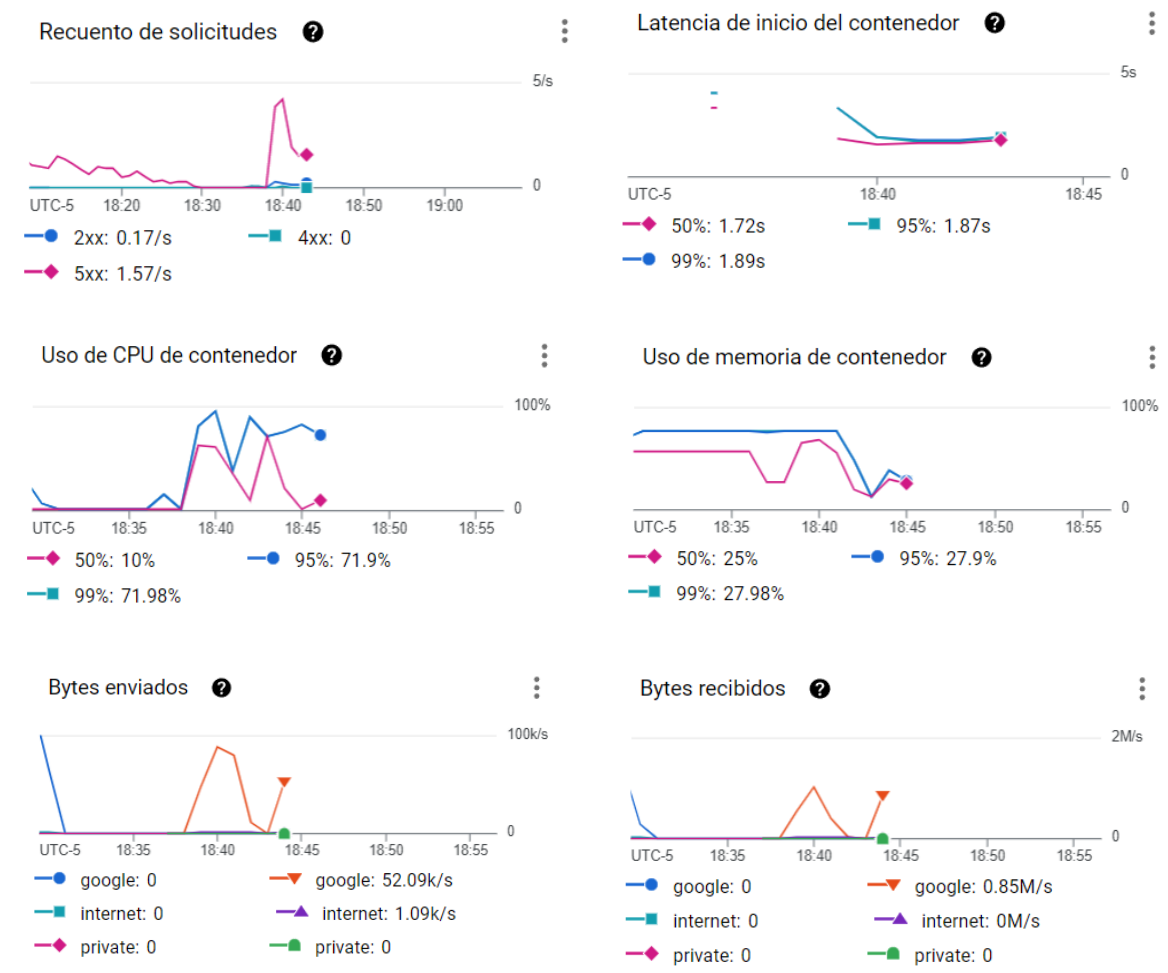
El número de solicitudes atendidas por segundo fue mucho menor al obtenido anteriormente en las pruebas de la capa web. Esto tiene sentido teniendo en cuenta que la tarea de compresión de archivos es más demandante para la aplicación.

Cloud Run - Worker

Este fue el componente que tuvo un mayor impacto durante las pruebas de carga teniendo en cuenta las limitaciones de hardware definidas: 512 MB de memoria, 1 CPU. Adicionalmente se definió un máximo de 3 instancias. En la siguiente imagen se puede observar el error obtenido por la limitante de hardware:



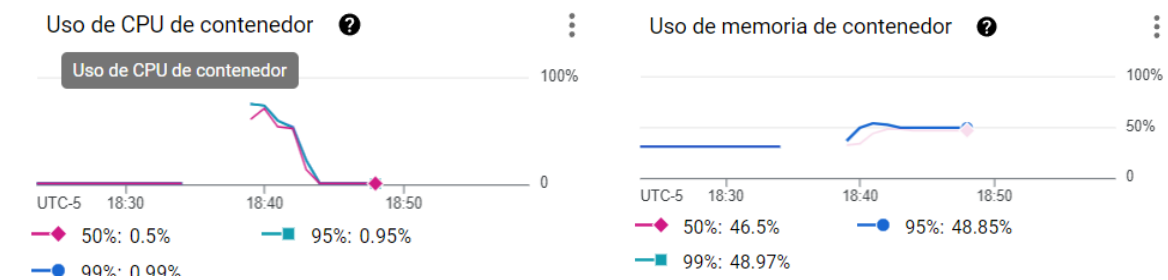
Adicionalmente, en las siguientes gráficas se pueden observar el comportamiento de algunas medidas tomadas durante la ejecución de la prueba. Un punto importante por resaltar es que la prueba llegó a ocupar el 100% de uso para la CPU. Asimismo, la memoria usada estuvo en alrededor del 85%, lo que tiene sentido para los errores recibidos durante la ejecución.

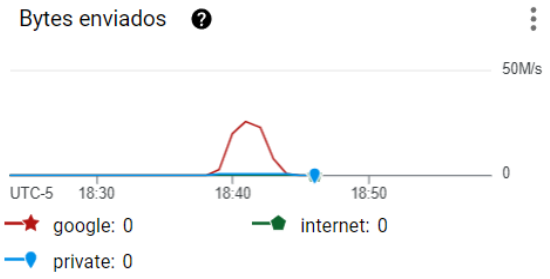


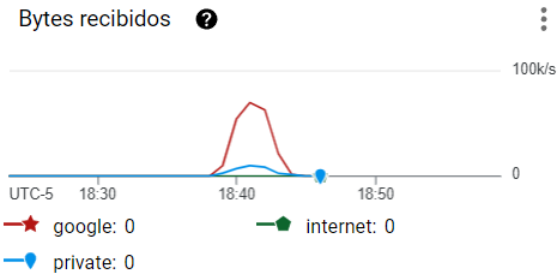
Finalmente, se puede observar el número de solicitudes simultáneas para los 12 usuarios concurrentes usados en la prueba:

Cloud Run - Web Sever

La capa web se usó únicamente para el inicio de sesión de los usuarios previamente creados. Como se puede observar en las gráficas analizadas para este componente, no hubo limitantes a la hora del procesamiento.

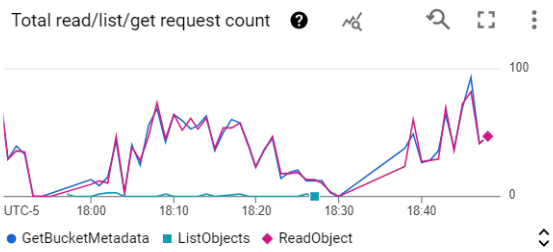


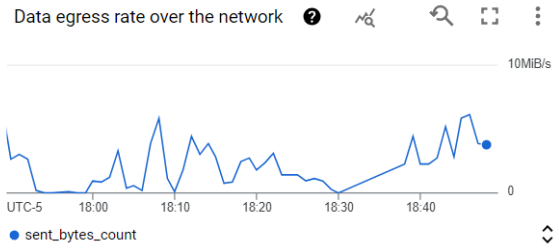




Cloud Storage - Bucket

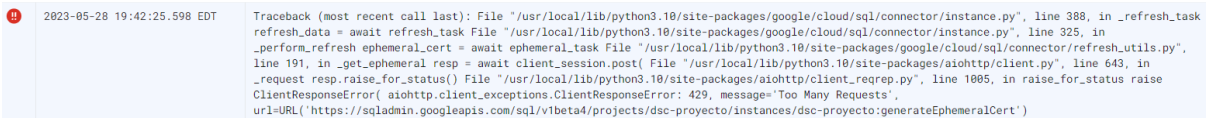
Este componente tuvo un buen desempeño almacenando los archivos sin procesar y procesados, y tuvo una carga de lectura y escritura en este escenario. El número de megabytes de ingreso fue mucho mayor a el número de megabytes de salida.



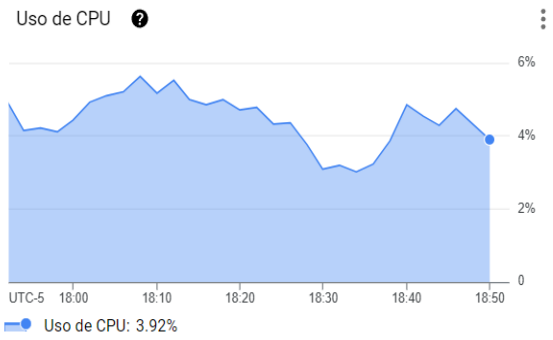


Cloud SQL

Este componente se vio altamente impactado por las solicitudes de consulta realizada por el Worker para traer algunos de los datos de los usuarios, y con ellos poder terminar el procesamiento satisfactoriamente. Debido al alto número de solicitudes, se observó un limitante del número de consultas que se pudieron realizar, como se puede ver en la siguiente imagen:



El uso de CPU de este componente fue bajo a diferencia del número de consultas realizadas hacia este. Sin embargo, se alcanzó el límite de operaciones de escritura como se observa en la imagen:



Archivos comprimidos en Bucket

Tras la prueba de carga se pudo confirmar el procesamiento de archivos en el Bucket creado en Cloud Storage, en donde en la siguiente imagen se puede observar los diferentes archivos y formatos usados para la compresión:

Nombre	Tamaño	Tipo	Fecha de creación	Clase de almacenamiento	Última modificac
TestFile92.7z	24.2 KB	application/x-7z-compressed	28 may 2023 17:47:39	Standard	28 may 2023 1
TestFile922.7z	24.2 KB	application/x-7z-compressed	28 may 2023 17:49:05	Standard	28 may 2023 1
TestFile937.zip	1.2 MB	application/zip	28 may 2023 17:47:11	Standard	28 may 2023 1
TestFile94.zip	1.2 MB	application/zip	28 may 2023 17:43:58	Standard	28 may 2023 1
TestFile951.zip	1.2 MB	application/zip	28 may 2023 17:49:14	Standard	28 may 2023 1
TestFile954.zip	1.2 MB	application/zip	28 may 2023 17:49:15	Standard	28 may 2023 1
TestFile955.7z	24.2 KB	application/x-7z-compressed	28 may 2023 17:49:16	Standard	28 may 2023 1
TestFile957.zip	1.2 MB	application/zip	28 may 2023 17:49:14	Standard	28 may 2023 1
TestFile958.7z	24.2 KB	application/x-7z-compressed	28 may 2023 17:51:12	Standard	28 may 2023 1

5. Conclusiones

A partir del desarrollo de este proyecto se comprueba como el rendimiento de la aplicación mejora al separar la capa de procesamiento y almacenamiento, lo cual permite el uso de componentes transversales para la persistencia de los datos y el uso de herramientas que se encargan de escalar automáticamente las tareas de procesamiento, como por ejemplo Cloud Run.

En las versiones anteriores de la aplicación web se pudo comprobar su funcionamiento, restringido a limitantes de capacidad fija de recursos o la necesidad de configurar auto balanceadores y garantizar la actualización de las instancias de Compute Engine. Con esta nueva versión de la aplicación que hace énfasis en el uso de PaaS fue posible mantener las ventajas del auto escalamiento, y concentrar esfuerzos en mejorar la lógica del negocio, y desentenderse de temas de infraestructura (tipo de máquina, tamaño de las máquinas, versión del S.O, parches, entre otros) y de algunos temas de seguridad. Sin embargo, el perder el control sobre la infraestructura plantea retos como lo es el generar desarrollos que busquen ser más eficientes en términos de procesamiento y memoria, dadas las limitaciones de las instancias de Cloud Run. También, se evidencia la necesidad de monitorear las métricas de uso para establecer con mayor precisión los límites del auto escalamiento que permiten aprovisionar los recursos necesarios para mantener la disponibilidad y rendimiento del sistema que espera el usuario, sin impactar negativamente los costos de las organizaciones. Por ejemplo, para este proyecto desarrollado se podría decir que no es necesario aumentar las instancias para el servidor web, pero sí que se hace necesario para el “worker” que comprime los archivos; y se puede hacer esta recomendación tras estudiar el comportamiento de ambos componentes ante escenarios de un aumento en las solicitudes.

Por último, se debe tener en cuenta que para aprovechar mejor las ventajas que ofrecen las arquitecturas Cloud existen condiciones que deben examinarse como lo son el separar las funcionalidades de acuerdo con el objetivo que realizan, el ver el almacenamiento y seguridad como elementos transversales, el definir límites, alertas y un presupuesto de los recursos alquilados para que sean rentables especialmente al considerar la facilidad con la que se puede aumentar la cantidad de recursos de un proyecto; el realizar pruebas y monitorear cómo se comporta el sistema en distintos escenarios para minimizar el riesgo de caídas en el sistema, el evaluar el uso de herramientas de terceros y las restricciones que impone y que afectan el rendimiento y por tanto, el éxito de los propios servicios. Ajustar las soluciones a los requerimientos planteados y a las particularidades de la nube permite una escalabilidad exitosa, que se adapta eficientemente a la demanda de usuarios finales, y optimizar los costos de las empresas.

6. Anexos

6.1. Código de configuración del Cloud Run y Pub Sub

```
#CLOUD RUN
#Clonación del repositorio
sudo git clone -b entrega_final https://github.com/aguerrero10/Desarrollo-cloud-grupal.git

#Ubicación en ruta del servidor web ------------------------
cd Desarrollo-cloud-grupal/WebServer/flaskr
gsutil cp gs://proyecto-dsc/resources/dsc-proyecto-b6565f206c96.json dsc-proyecto-b6565f206c96.json

#Creación de la imagen del servidor web
gcloud builds submit --tag gcr.io/dsc-proyecto/compress-files

#Despliegue de la imagen del servidor web
gcloud run deploy compress-files --image gcr.io/dsc-proyecto/compress-files \
  --platform managed --region us-central1 --allow-unauthenticated --max-instances=3

#Ubicación en ruta del Worker -----------------------------
cd Desarrollo-cloud-grupal/Worker

#Creación de la imagen del Worker
gcloud builds submit --tag gcr.io/dsc-proyecto/worker-files1

#Despliegue de la imagen del Worker
gcloud run deploy worker-files1 --image gcr.io/dsc-proyecto/worker-files1 \
  --platform managed --region us-central1 --allow-unauthenticated --max-instances=3

#CUENTA DE SERVICIO
#Creación de cuenta de servicio para ejecutar Cloud Run
gcloud iam service-accounts create cloud-run-pubsub-invoker \
    --display-name "Cloud Run Pub/Sub Invoker"

#Adición de permisos a cuenta de servicio
gcloud run services add-iam-policy-binding worker-files \
--member=serviceAccount:cloud-run-pubsub-invoker@dsc-proyecto.iam.gserviceaccount.com \
--role=roles/run.invoker

#Permitir la creación de Tokens
gcloud projects add-iam-policy-binding dsc-proyecto \
   --member=serviceAccount:service-357876382465@gcp-sa-pubsub.iam.gserviceaccount.com \
   --role=roles/iam.serviceAccountTokenCreator

#PUB-SUB
#Configuración del publicador para notificar cada vez que se cargue un nuevo archivo
gsutil notification create -f json -t tareas -p files/uploads/ -e OBJECT_FINALIZE gs://proyecto-dsc

#Configuración del suscriptor para ejecutar el contenedor de Cloud Run
gcloud pubsub subscriptions create myWorkerSubs --topic tareas \
--ack-deadline=600 \
--push-endpoint=https://worker-files1-ysnuqmb7aa-uc.a.run.app/ \
--push-auth-service-account=cloud-run-pubsub-invoker@dsc-proyecto.iam.gserviceaccount.com
```