ENTREGA 3 - DESPLIEGUE BÁSICO EN LA NUBE
MIGRACIÓN DE UNA APLICACIÓN WEB A LA NUBE PÚBLICA

Danny Pineda Echeverri, Felipe Alejandro Paez Guerrero, Vihlai Maldonado Cuevas
ISIS4426 - Desarrollo de soluciones cloud
Universidad de los Andes, Bogotá, Colombia
{d.pinedae, f.paezg, v.maldonado1}@uniandes.edu.co
Fecha de presentación: abril 2 de 2023

1. Arquitectura de la aplicación

En la presente entrega se realizó la adaptación del código y el despliegue sobre la nube de Google. Para ello se realizó uso de Compute Engine, Cloud SQL y VPC.

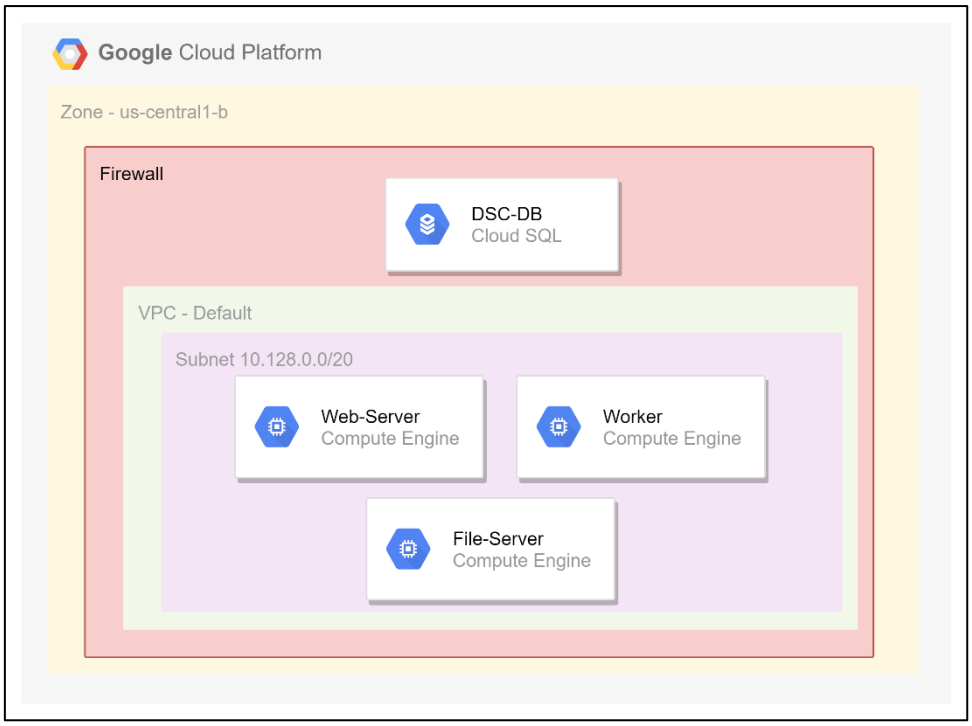


Figura 1. Arquitectura de despliegue GCP.

Donde

File-Server:

- Zona: us-central1-b
- Tipo de máquina: f1-micro
    - Memoria: 614 MB
    - Disco: 10GB
    - CPU: 1 vCPU

Web-Server:

- Zona: us-central1-b
- Tipo de máquina: f1-micro
    - Memoria: 614 MB
    - Disco: 10GB

- CPU: 1 vCPU

Worker:

- Zona: us-central1-a
- Tipo de máquina: n1-highcpu-2
  - Memoria: 1.8 GB
  - Disco: 10GB
  - CPU: 2 vCPU

El worker se ajustó para tener 2 vCPU debido a que hubo un problema respecto al despliegue del contenedor de RabbitMQ y Celery.

Para la conectividad se estableció la siguiente topología:

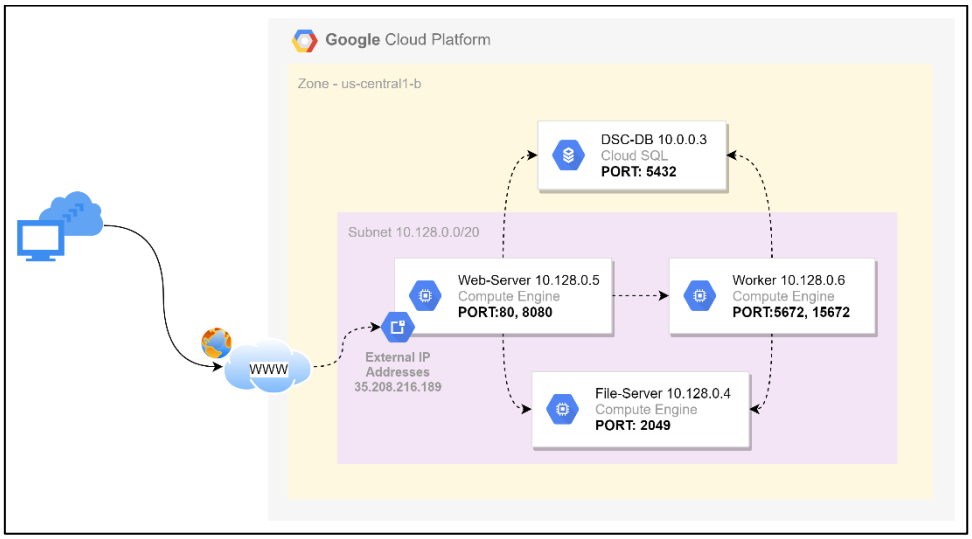


Figura 2. Topología de Red GCP.

La base de datos DSC-DB se desplegó en la misma zona de las máquinas virtuales, tomo la IP 10.0.0.3 y el servicio se expuso sobre el puerto 5432

Las maquinas Web-server, Worker y File-Server se configuraron para tener las IPs privadas fijas 10.128.0.5, 10.128.0.6 y 10.128.0.4 respectivamente dentro de la subnet 10.128.0.0/20. Para la VM web server se expuso los puertos 80 y 8080 sobre la IP publica estática 35.208.216.189 garantizando que esta no va a cambiar después de encender o reiniciar la VM. El Worker expone los puertos 5672 y 15672 y el File-Server exponen el puerto 2049.

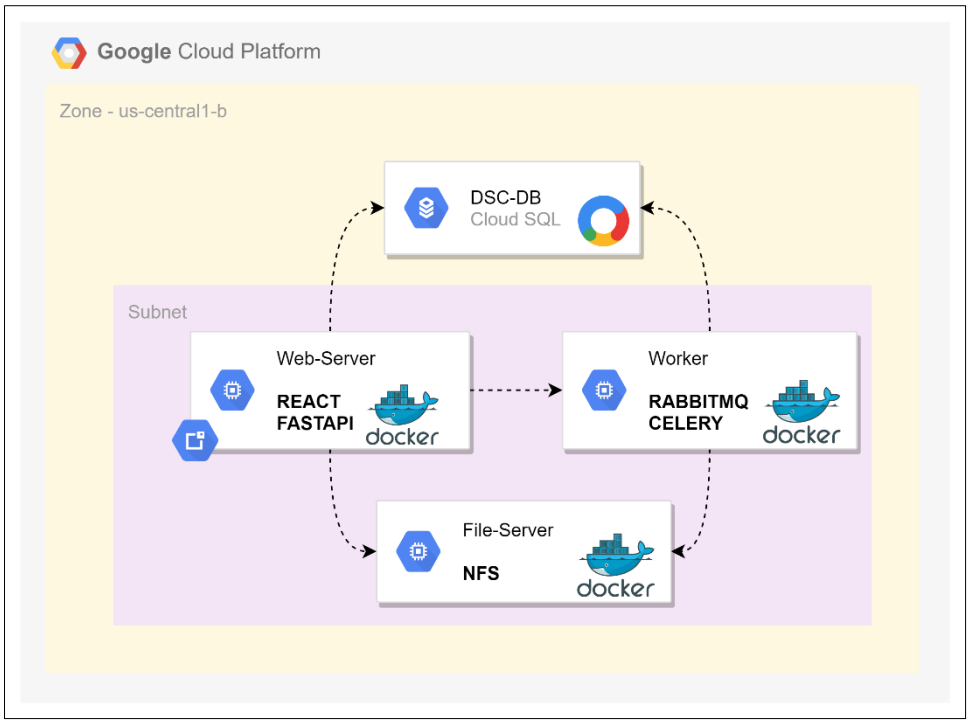
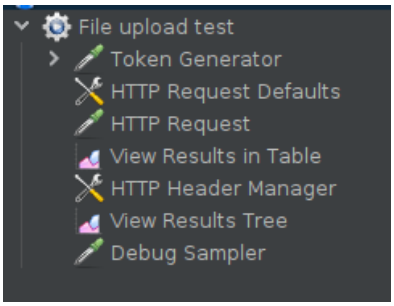


Figura 3. Topología de Despliegue componentes de aplicación GCP.

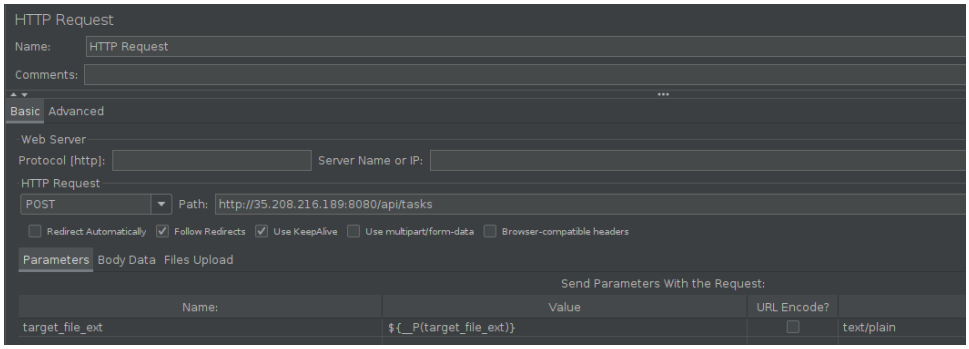
2. Pruebas de estrés

Con el fin de dimensionar la capacidad de la aplicación y la infraestructura de soporte se realizaron pruebas de estrés utilizando la herramienta JMeter con los escenarios de prueba definidos en la entrega 2.

JMeter[1] es un software libre que permite realizar pruebas de rendimiento de recursos y aplicaciones web estáticos y dinámicos. Puede ser utilizado para simular carga pesada sobre un servidor, grupo de servido res, redes u objetos para probar su resistencia o para analizar su rendimiento general bajo diferentes tipos de carga.



De acuerdo con la figura anterior se creó un *Thread Group* en Jmeter para simular la carga de los usuarios. Tenemos el *Token Generator* encargado de obtener el *Bearer Token* de la aplicación y guardarlo en una variable para ser usado por los demás *endpoints* y el *Http Request* donde se define el envío de las solicitudes de compresión de archivos.



En la figura anterior tenemos la configuración hecha para el envío de los archivos, donde el target_file_ext se dejó como parámetro con el fin de poder cambiarlo para las diferentes pruebas.

2.1. Escenario 1

El primer escenario considerado busca identificar la máxima cantidad de requests HTTP por minuto que soporta la aplicación web. Para ello se considera:

Pocos usuarios enviando archivos pequeños:

De 1 a 100 usuarios concurrentes enviando archivos pequeños (alrededor de 15 MB).

Tipo de gráfica: Gráfico de líneas, con cantidad de usuarios concurrentes en el eje x y las métricas en el eje y.

Se ejecutarán las pruebas desde una máquina de AWS Academy, con el fin de simular el acceso de los usuarios desde diferentes ubicaciones.

Restricciones:

Tiempo de respuesta máximo: 30 segundos.

Máxima tasa de error: 1%

Cantidad máxima de usuarios concurrentes: 100

Asimismo, cabe resaltar que durante las pruebas se registraron las métricas y tiempos de procesamiento de los archivos que se alcanzaron a procesar previo a la interrupción del servicio. Esto debido a que algunas de las instancias interrumpieron su servicio debido al consumo de memoria o procesamiento durante las pruebas.

Resultados:

2.1.1. Compresión Zip

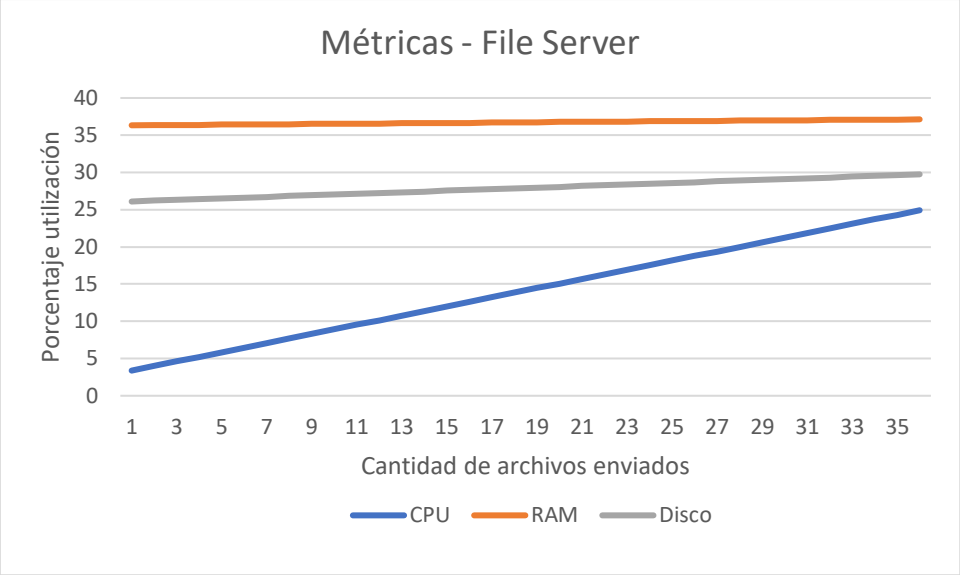


Figura 4. Métricas de los recursos de File Server.

La figura Figura 4. Métricas de los recursos de File Server.muestra el consumo de CPU, memoria y disco de la instancia dedicada para el File Server. Se observa que no hubo interrupción en el servicio.

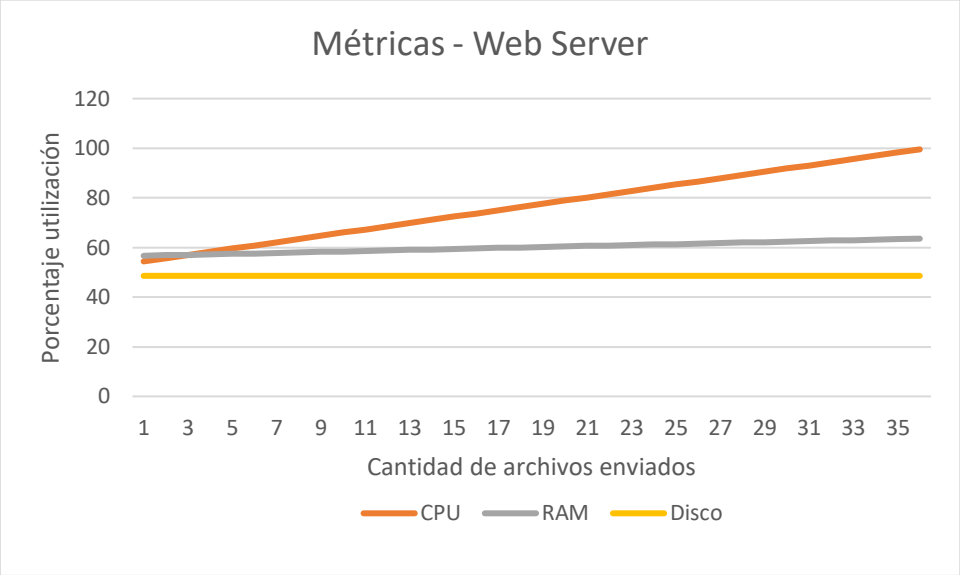


Figura 5. Métricas de los recursos de Web Server.

La Figura 5. Métricas de los recursos de Web Server. muestra el consumo de CPU, memoria y disco de la instancia dedicada para el Web Server. Se observa que hubo interrupción en el servicio ya que la utilización de la CPU aumentó al 100% en el archivo 35.

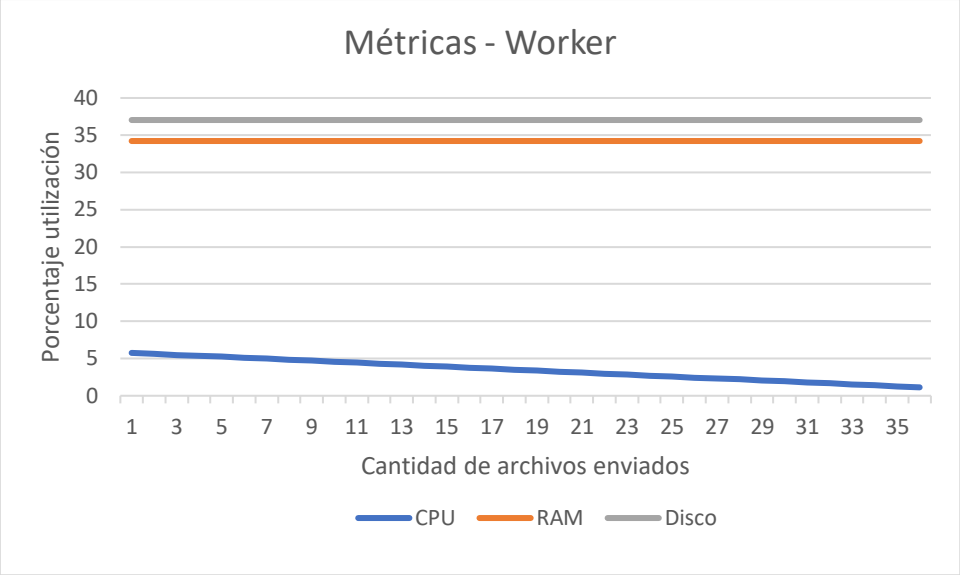


Figura 6. Métricas de los recursos de Worker.

La Figura 6. Métricas de los recursos de Worker. muestra el consumo de CPU, memoria y disco de la instancia dedicada para el worker. Se observa que no hubo interrupción en el servicio.

En resumen para la prueba con compresión tipo zip se pudieron procesar hasta 35 archivos de 100 en total ya que la instancia del Web Server se detuvo debido a un incremento en el consumo de la CPU.

2.1.2. Compresión Tar.gz

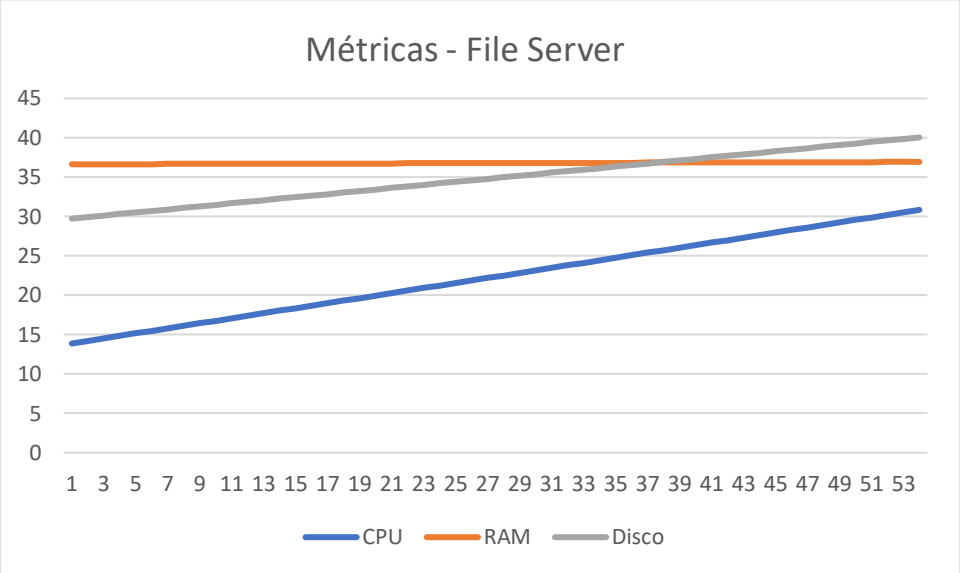


Figura 7. Métricas de los recursos de File Server.

La Figura 7. Métricas de los recursos de File Server. muestra el consumo de CPU, memoria y disco de la instancia dedicada para el File Server. Se observa que no hubo interrupción en el servicio.

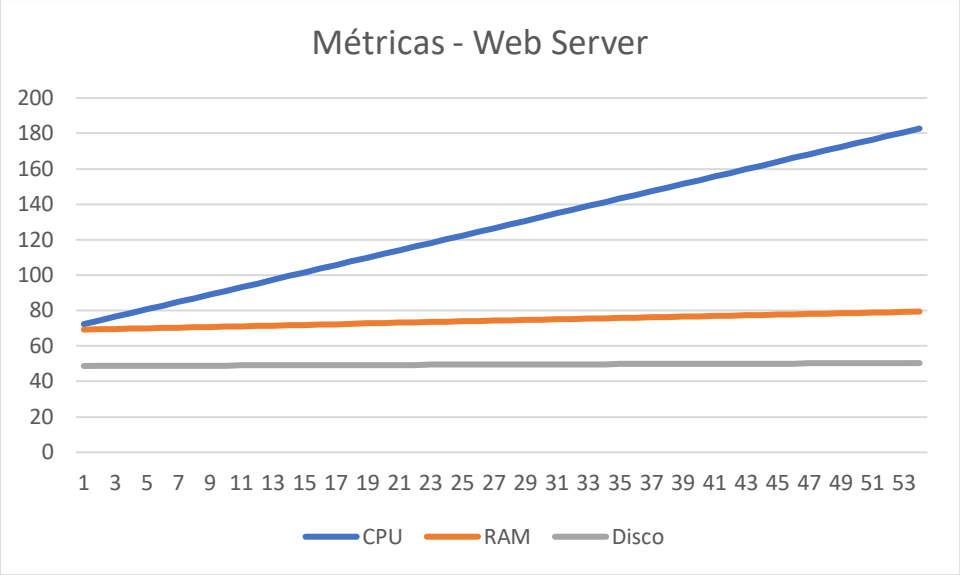


Figura 8. Métricas de los recursos de Web Server.

La Figura 8. Métricas de los recursos de Web Server. muestra el consumo de CPU, memoria y disco de la instancia dedicada para el Web Server. Se observa que hubo interrupción en el servicio ya que la utilización de la CPU aumentó al 100% en el archivo 53.

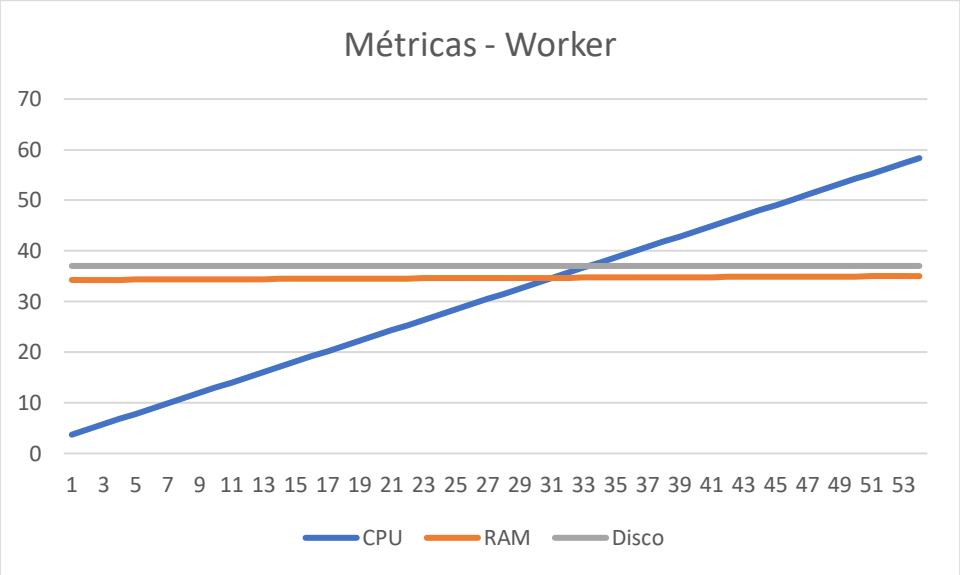


Figura 9. Métricas de los recursos de Worker.

La Figura 9. Métricas de los recursos de Worker. muestra el consumo de CPU, memoria y disco de la instancia dedicada para el worker. Se observa que no hubo interrupción en el servicio.

En resumen para la prueba con compresión tipo zip se pudieron procesar hasta 53 archivos de 100 en total ya que la instancia del Web Server se detuvo debido a un incremento en el consumo de la CPU.

2.2. Escenario 2

El segundo escenario busca identificar la máxima cantidad de archivos que pueden ser procesados por minuto en la aplicación local.

Restricciones:

Tamaño mínimo de archivos: 10MB.

Espera máxima: 30 segundos.

Capacidad de procesamiento mínima: carga de 90 archivos por minuto.

Mínima tasa de transferencia de datos: 75MB por segundo de subida.

Resultados:

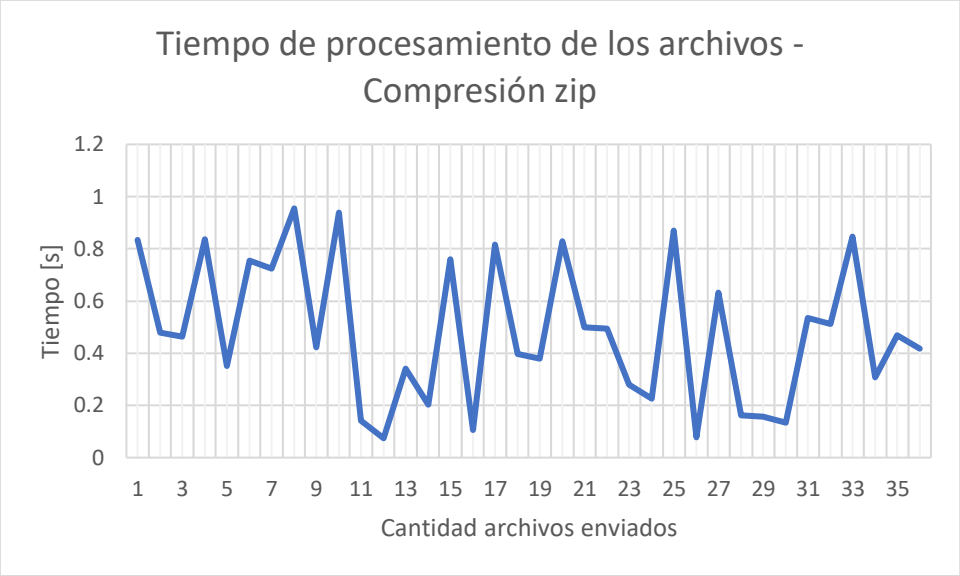


Figura 10. Tiempo de procesamiento de los archivos con compresión zip.

En la Figura 10. Tiempo de procesamiento de los archivos con compresión zip. se grafica el tiempo de procesamiento de los archivos con compresión zip. El tiempo medio es de 0,484 segundos. El tiempo máximo de procesamiento es de 0.95 segundos.

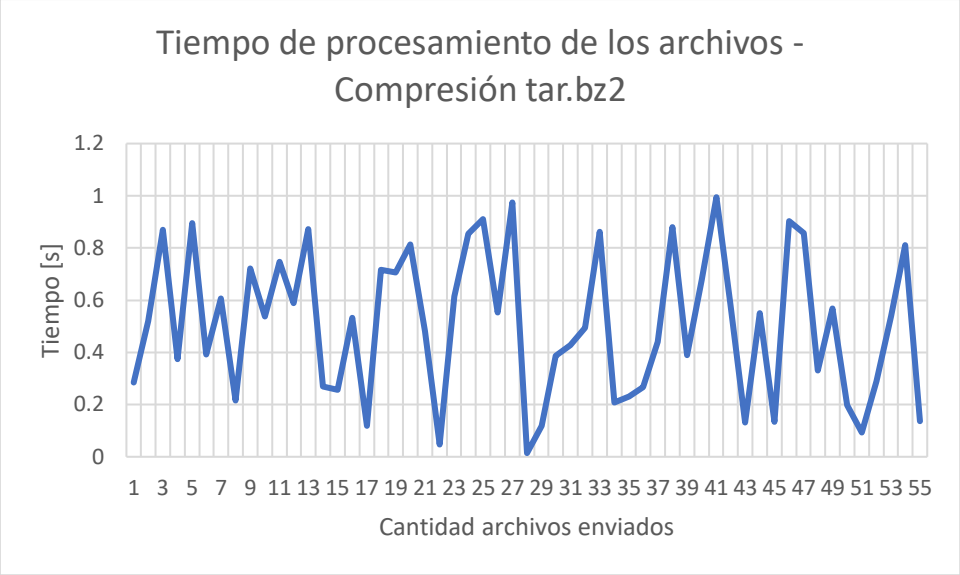


Figura 11. Tiempo de procesamiento de los archivos con compresión tar.bz2.

En la Figura 11. Tiempo de procesamiento de los archivos con compresión tar.bz2. se observa el tiempo de procesamiento de los 55 archivos con compresión tar.bz2. El tiempo medio es de 0,508 segundos. El tiempo máximo de procesamiento es de 0,994 segundos.

En general se observa que el tiempo de procesamiento para el tipo de compresión tar.bz2 es más alto que él .zip, sin embargo, se alcanzaron a procesar 55 archivos para compresión tar.bz2 y 35 con .zip.

3. Consideraciones

En cuanto a la aplicación tenemos:

Formato de entrada permitidos: No hay restricciones de formato de entrada.

Formatos de compresión permitidos: ZIP, TAR.GZ, TAR.BZ2

Requerimientos de seguridad de la contraseña al momento de solicitar la creación del usuario:

- La contraseña debe contener más de 8 caracteres.
- La contraseña debe contener caracteres especiales.
- La contraseña debe contener un número.
- La contraseña debe contener una letra mayúscula.

Para escalar la aplicación se propone:

- La instancia del Worker que contiene los contenedores de RabbitMQ y Celery se debe aprovisionar con mayor capacidad de memoria ya que con 614MB de RAM únicamente no fue posible iniciar los contenedores.
- Para hacer el despliegue del frontend se optó por escalar la instancia del Web Server con mayor memoria de tal manera que pudiera generar los archivos estáticos a través

de React y luego se disminuyó su memoria a 614 MB. Esto debido a que sólo con 614 MB no fue posible construir la aplicación y hacer el despliegue en el contenedor.

- De acuerdo con los resultados del escenario uno, se evidenció un alto consumo de CPU por parte del Web Server, limitando así la finalización de las pruebas. Para poder escalar a cientos de usuarios se propone contar con más instancias para los Worker, es decir escalar de manera horizontal.
- De acuerdo con los resultados del escenario dos, en los tiempos de procesamiento obtenidos para los dos tipos de compresión escogidos, estos fueron menores a 1 segundo. Dado que el tiempo máximo de espera propuesto es de 30 segundos, se concluye que para escalar se deben agregar más instancias de Workers, pero la instancia seleccionada es capaz de procesar los archivos dentro de los tiempos adecuados.

4. CONCLUSIONES

- Se adaptó la aplicación que permite la compresión de archivos a través de un sistema de encolamiento entre Celery y RabbitMQ para que fuera posible desplegarla en la nube de Google cloud. Esta aplicación permite una mayor escalabilidad y mejor tolerancia a fallos ya que las tareas son procesadas de manera asíncrona y distribuida, permitiendo a varios usuarios acceder y utilizar la aplicación de manera concurrente.
- La dockerización de la aplicación permitió agilizar el despliegue de los componentes en las instancias y se enfocó más en los archivos de configuración y la conectividad de estas.

5. Referencias

[1] Apache JMeter , URL: https://jmeter.apache.org/