ENTREGA 4 - 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: mayo 7 de 2023

PARTE A - MODELO DE DESPLIEGUE - ESCALABILIDAD EN LA CAPA WEB

1. Modelo de despliegue

A continuación, se define la topología de red para definir la escalabilidad de la capa web.

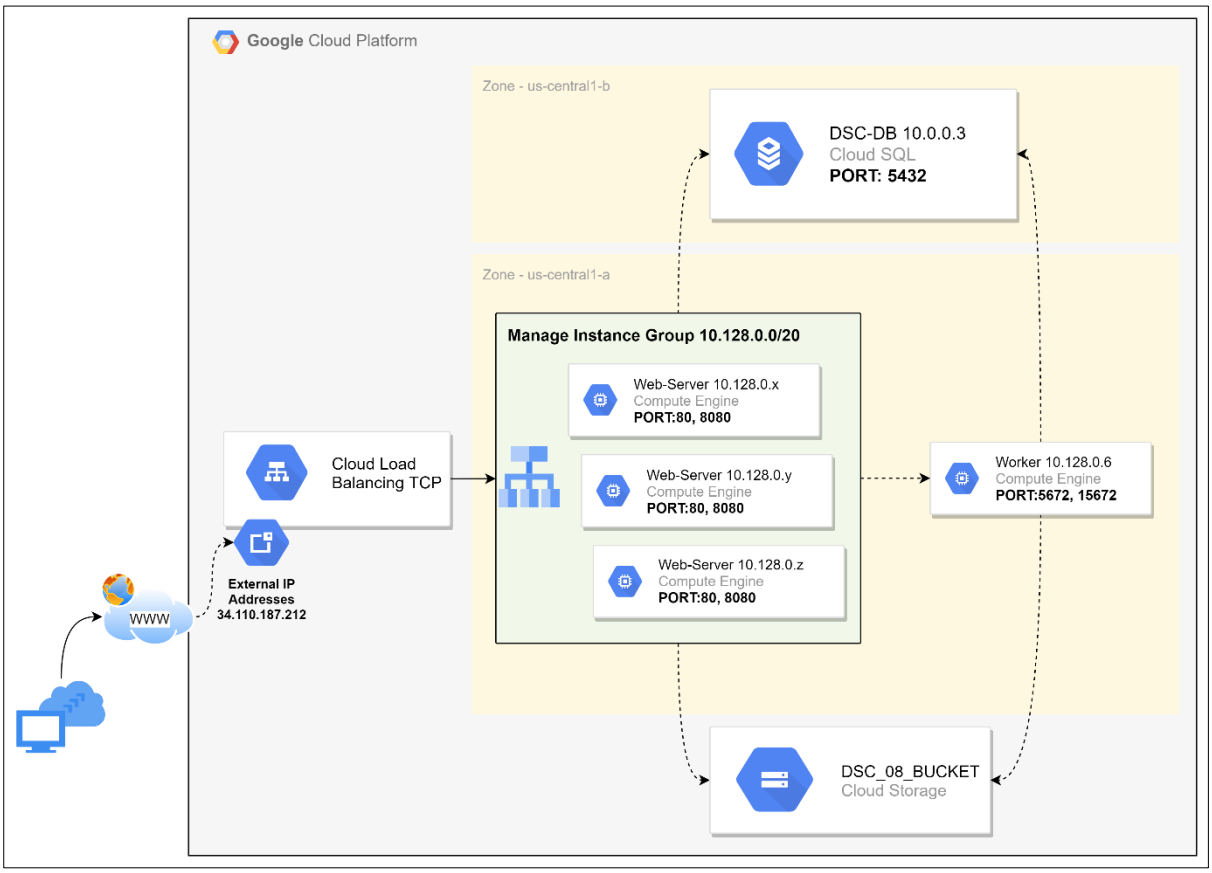

Figura 1. Arquitectura Conectividad GCP-Escalabilidad Web.

Para esto se definieron los siguientes elementos:

- Un Instance Template con ejecución de script de inicio.
- Un MIG (Manage Instance Group) que usa el Instance Template anterior para ejecutar eventos de ScaileIn y ScaleOut.
- Un balanceador de carga externo TCP con una IP de Front Publica respondiendo por puertos 80 y 8080 y el MIG para el Backend.
- Reglas de Firewall para la ejecución de Health Checks desde el balanceador.
- Bucket para el servicio de almacenamiento de archivos.

El Instance Template se creó usando un tamaño de maquina g1-small con SO Debian. La principal característica del Template es que se configuro en la Metadata la ejecución de un *startup-script* el cual contiene todas las instrucciones necesarias que se deben correr en cualquier maquina nueva (En la sección de anexos se puede observar el script completo).

Figura 2.1 Instances Template Imagen.

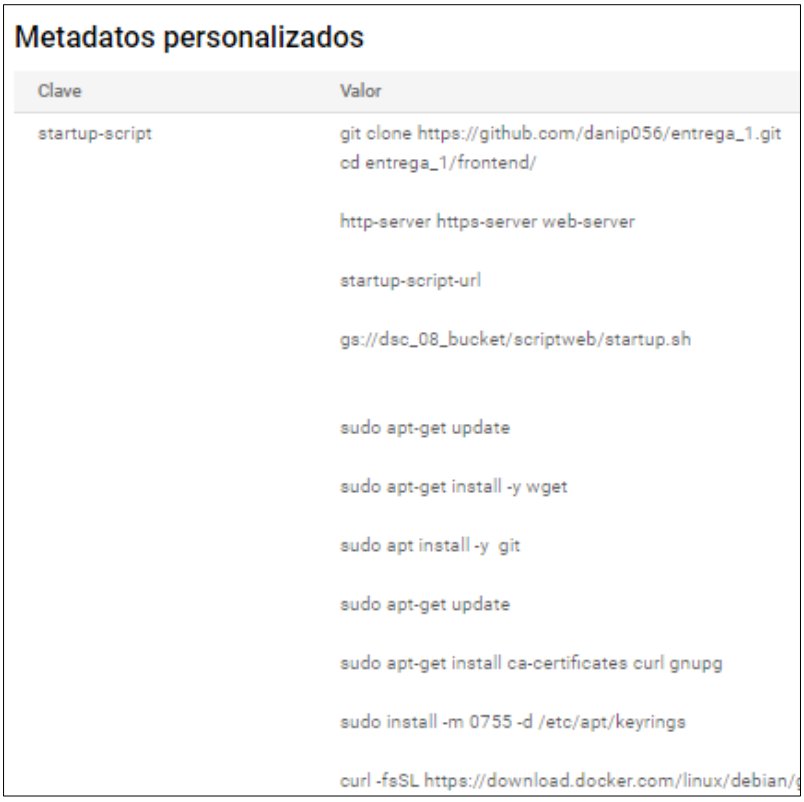

Figura 2.2 Instances Template Metadata.

Para el MIG se utilizó la plantilla anterior y que utilizara los parámetros de escalamiento que se explicaran en el punto 2 parte A.

Figura 3 Manage Instance Group Web MIG-Web.

Para el balanceo de cargas se crearon reglas de firewall para que una VM aceptara conexiones TCP desde los rangos de IPs 130.211.0.0/22 and 35.191.0.0/16.

Figura 4. Reglas de Firewall para HC del LB.

De esta manera se garantiza que el Load Balancer (LB) puede determinar si una VM puede recibir tráfico. Una vez se garantiza esto, se definió que el LB empezaría a realizar balanceo sobre las VM del MIG con el uso del algoritmo Maglev.

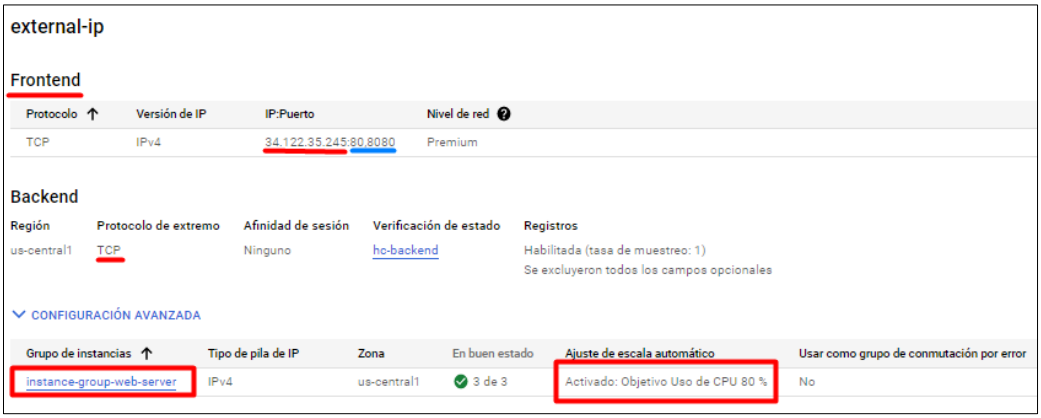

Figura 5. Configuración Load Balancer.

Se creo la IP Publica 34.110.187.212 como recurso de Frontend para la publicación y recepción de solicitudes a la aplicación. Se definió que recibiera solicitudes por puerto 80 (web) y puerto 8080 (Backend) para que sean balanceadas al MIG *instance-group-web-server*.

2. Estrategia de escalamiento

Dentro del MIG se definieron los valores y umbrales para los eventos de ScaleIn y ScaleOut:

- # VM: Min 1 – Max 3.
- Tipo de Trigger de AutoScaling: Utilización de CPU
- Evento de Autoscaling: 80% de utilización de CPU.

El último parámetro quiere decir que al llegar al 60% de la utilización de CPU, definida por el

algoritmo de Maglev, ocurrirá un evento de ScaleOut.

NOTA: Sobre este grupo se hicieron pruebas para que solo se desplegara en una zona de disponibilidad de acuerdo con la arquitectura de conectividad de la Figura 1. En la Parte B se realizaron los cambios necesarios para tener en cuenta la alta disponibilidad en zona.

3. Configuración de Bucket de almacenamiento:

Se determinó la creación del Bucket gs://dsc_08_bucket en Cloud Storage para el almacenamiento de archivos.

Figura 6 Configuración del Bucket.

Se definió para que no fuera de acceso público y tuviera una llave de encriptado.

4. Requerimientos funcionales

El funcionamiento de la aplicación se describe en el video adjunto en el repositorio.

5. Pruebas de carga

Se realizaron las pruebas de carga definidas e implementadas en entregas anteriores.

5.1. 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.

5.1.1. Conversión ZIP

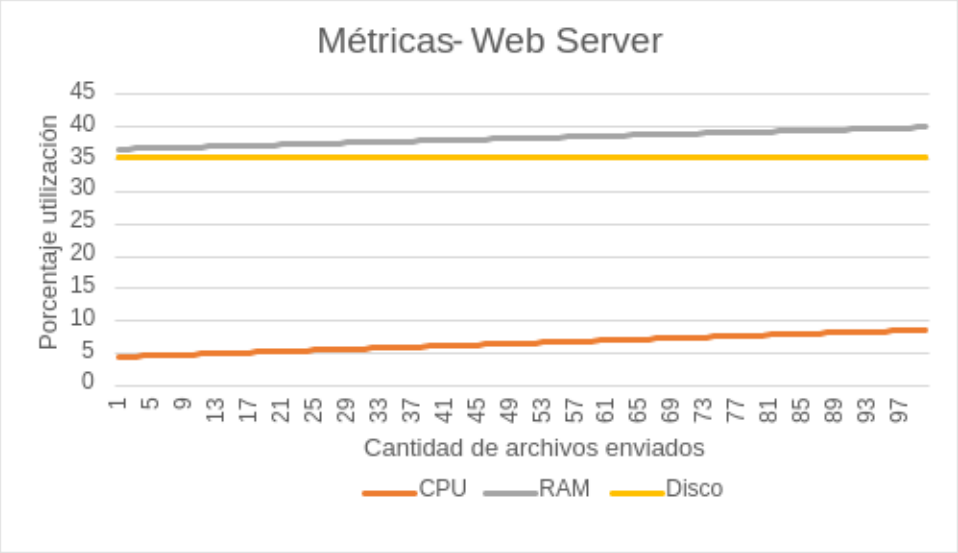

Figura 6. Métricas CPU, RAM y Disco Web Server conversión ZIP.

La Figura 6 muestra el consumo de CPU, memoria y disco de la instancia dedicada para el Web Server. Se observa que pudo manejar los 100 usuarios concurrentes, y en la consola se observó que no se escalaron las instancias.

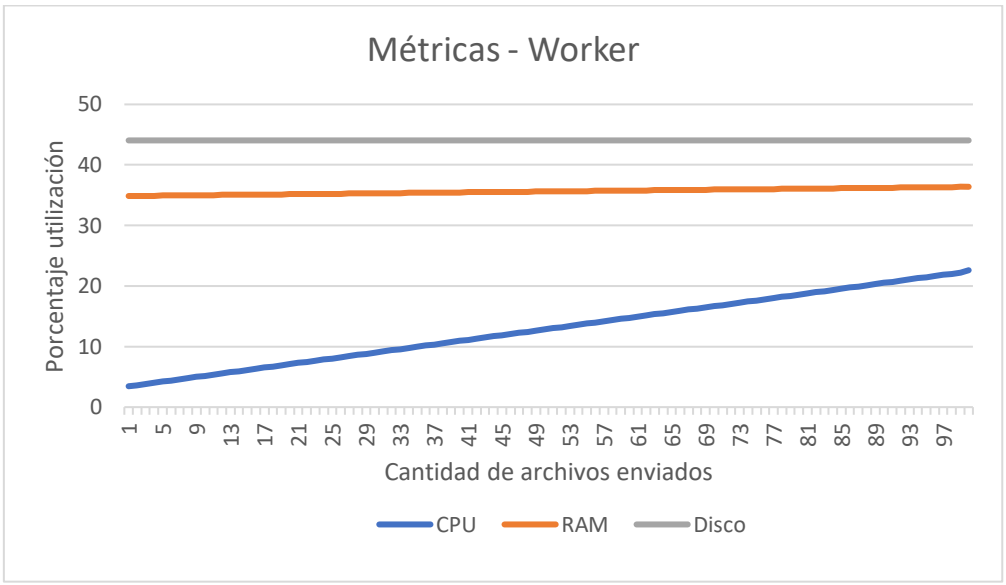

Figura 7. Métricas CPU, RAM y Disco Worker conversión ZIP.

La figura 7 muestra el consumo de CPU, memoria y disco de la instancia dedicada para el Worker. Se observa que hubo un incremento en la utilización de la CPU, sin embargo, el consumo de memoria se mantuvo constante.

En resumen, en la prueba de compresión tipo .zip se pudieron procesar los 100 archivos sin la necesidad de escalar el Web server. Asimismo, no hubo interrupción en el servicio.

5.1.2. Conversión Tar.bz2

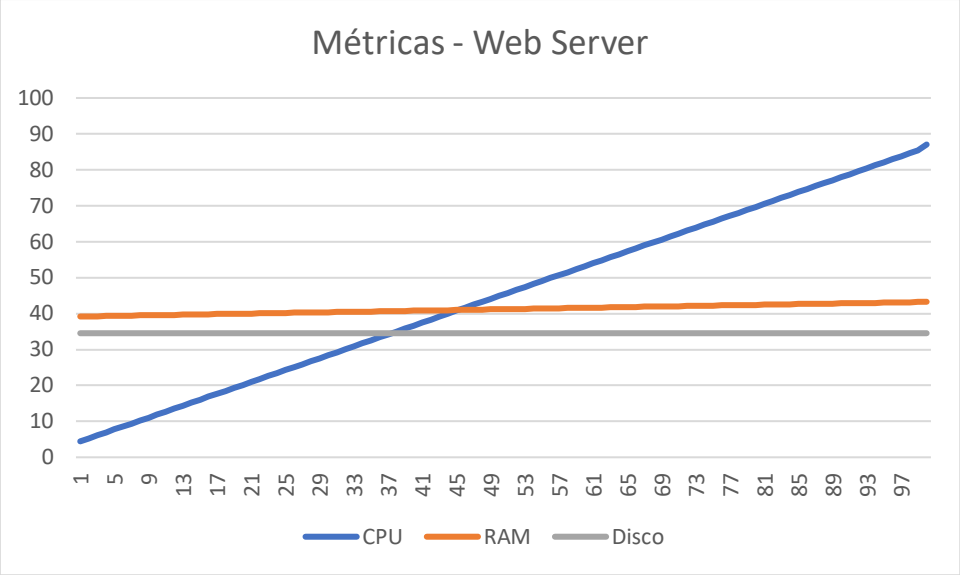

Figura 8. Métricas CPU, RAM y Disco Web Server conversión TAR.

La figura 8 muestra el consumo de CPU, memoria y disco del Web Server. Se observa que el consumo de CPU se incrementó hasta llegar cercano a 90%. El consumo de memoria se mantuvo por debajo del 50%. Dado que el grupo de auto escalado se fijó para escalar luego del 90% de utilización de CPU, no se llegó a escalar a otras instancias.

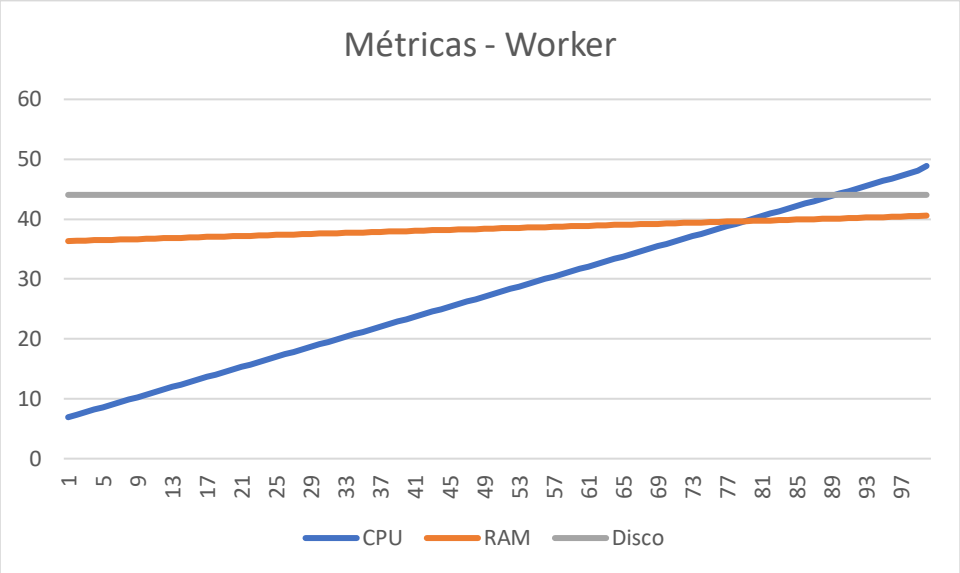

Figura 9. Métricas CPU, RAM y Disco Worker conversión TAR.

La figura 9 muestra el consumo de CPU, memoria y disco de la instancia dedicada para el Worker. Se observa que el consumo de CPU aumentó hasta el 50%, mientras el consumo de memoria se mantuvo cercano a 40%.

En resumen, para la prueba con compresión tipo tar.gz2 se pudieron procesar los 100 archivos y no se escaló el grupo de instancias del web server.

5.2 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:

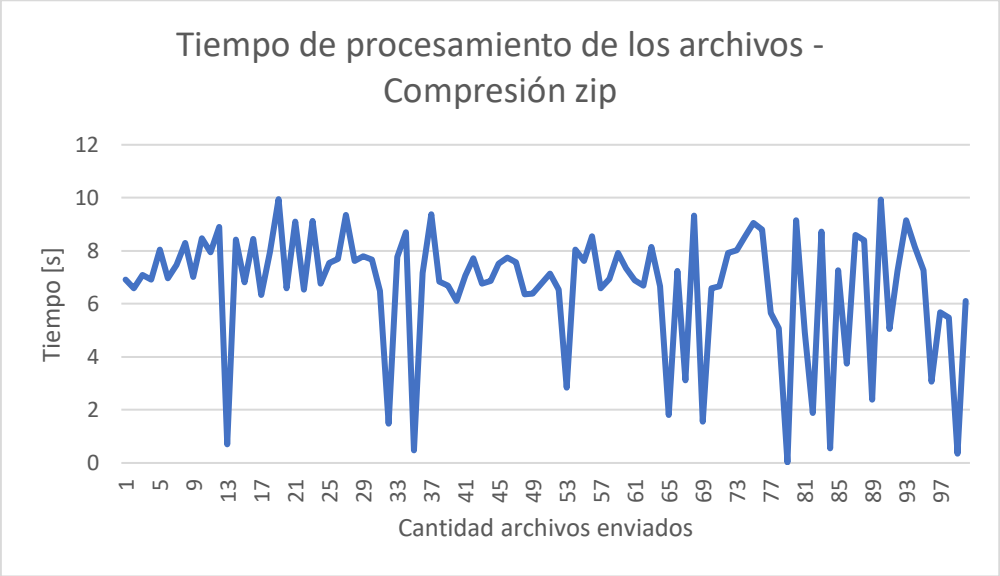

Figura 10. Tiempo Procesamiento ZIP.

En la **Error! Reference source not found.** se grafica el tiempo de procesamiento de los archivos con compresión zip. El tiempo medio es de 6.674 segundos. El tiempo máximo de procesamiento es de 9,95 segundos.

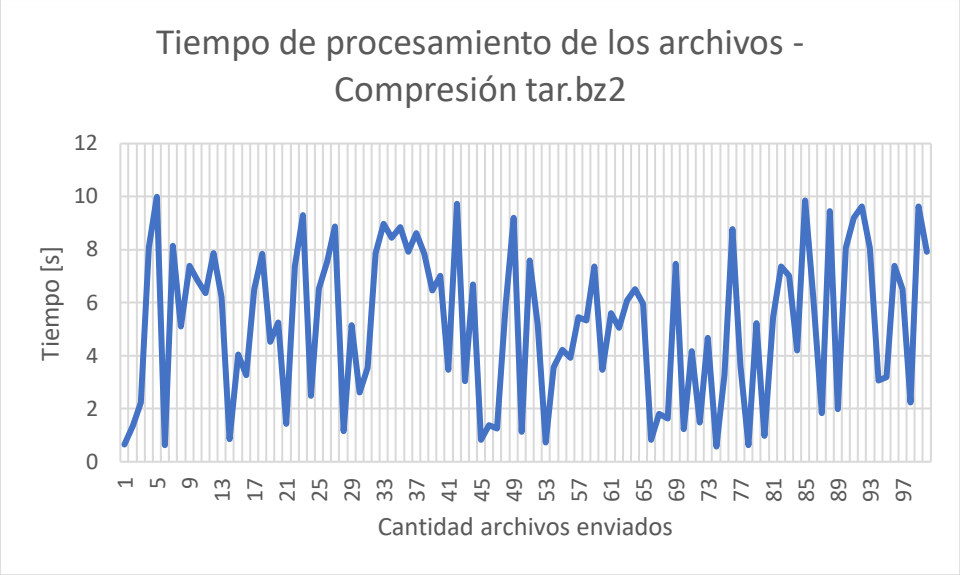

Figura 11. Tiempo Procesamiento TAR.

En la figura 11 se observa el tiempo de procesamiento de los 100 archivos con compresión tar.bz2. El tiempo medio es de 5.36 segundos. El tiempo máximo de procesamiento es de 9.99 segundos.

En general se observa que el tiempo de procesamiento para el tipo de compresión zip es más alto que él .tar.bz2, igualmente se alcanzaron a procesar los 100 archivos.

PARTE B - MODELO DE DESPLIEGUE - ESCALABILIDAD EN EL BACKEND

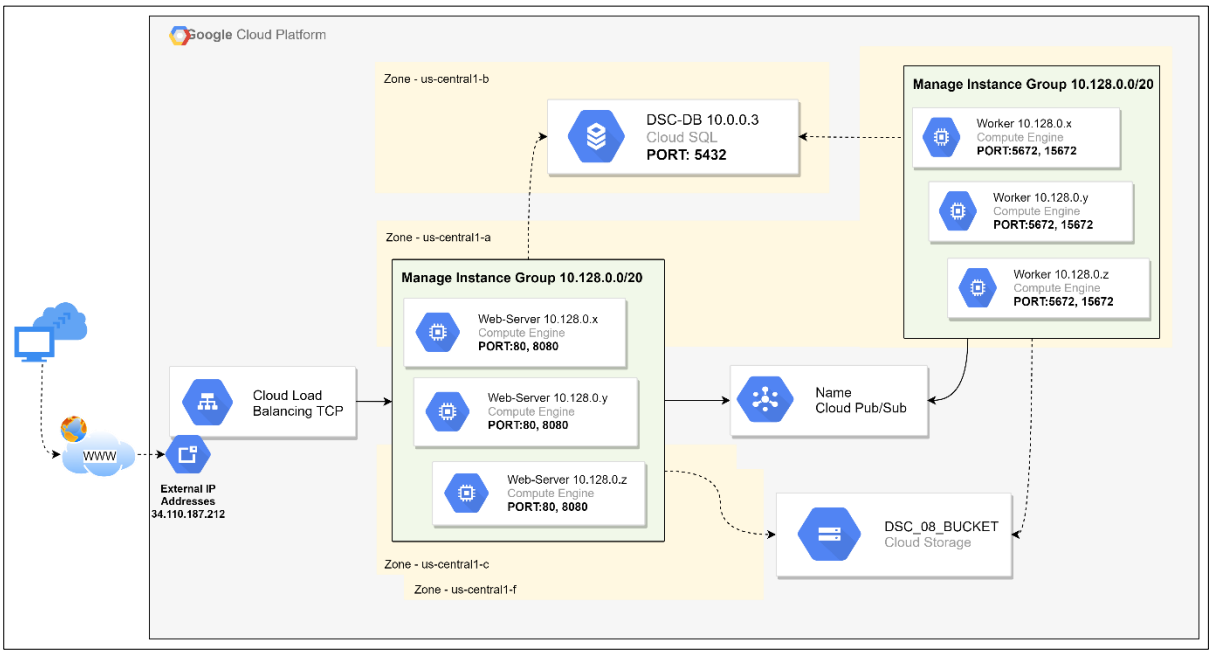

Figura 12. Arquitectura Conectividad GCP-Escalabilidad Worker.

En esta etapa se definieron los cambios necesarios para tener alta disponibilidad en zona para la capa web, mensajería utilizando el servicio de Pub/Sub y escalabilidad de los worker en una zona de disponibilidad.

Se cambio el parámetro de Multiple Zone en el MIG de la capa web dejando 3 zonas disponibles para la creación de VM.

Para la capa worker se utilizó un Instance Template usando un tamaño de maquina n1-highcpu-2 con SO Debian. La principal característica del Template es que se configuro en la Metadata la ejecución de un *startup-script* el cual contiene todas las instrucciones necesarias que se deben correr en cualquier maquina nueva (En la sección de anexos se puede observar el script completo).

6. Estrategia escalado automático de Workers.

Dentro del MIG se definieron los valores y umbrales para los eventos de ScaleIn y ScaleOut:

- # VM: Min 1 – Max 3.
- Tipo de Trigger de AutoScaling: Fila de Pub/Sub.
- Topic: Conversion request.
- Evento de Autoscaling: 4 mensajes pendientes por asignación por VM.

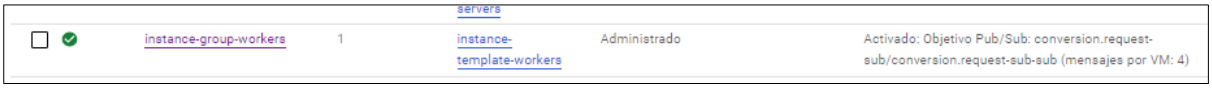

Figura 13. Manage Instance Group Worker MIG-Worker.

7. Configuración Cloud Pub/Sub

Se realizó la configuración del pub/sub con el fin de manejar a través de una cola de mensajes la creación de las tareas. Este fue el reemplazo de Celery y rabbitMQ.

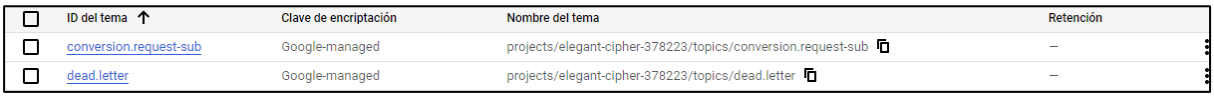

Figura 15. Tópicos creados en pub/sub.

En la figura 15 se observan los tópicos creados. Se configuró la suscripción de tal manera que manejara máximo 6 intentos con un delay exponencial.

8. Alta disponibilidad en Zonas para capa Web.

Para asegurar la alta disponibilidad por zona en la capa web, se creó un nuevo MIG cambio el parámetro Multizone en la zona us-central1-a/c/f como se observa en la figura a continuación:

Figura 14. Manage Instance Group configuración Multizona.

9. Pruebas de carga

A continuación, se describen los resultados obtenidos con las pruebas de carga en la parte B.

9.1. Escenario 1

Cabe resaltar que con la implementación multizona de las instancias del servidor web, se crearon 2 instancias adicionales en zonas distintas. En total se obtuvieron 3 instancias.

9.1.1. Compresión Zip

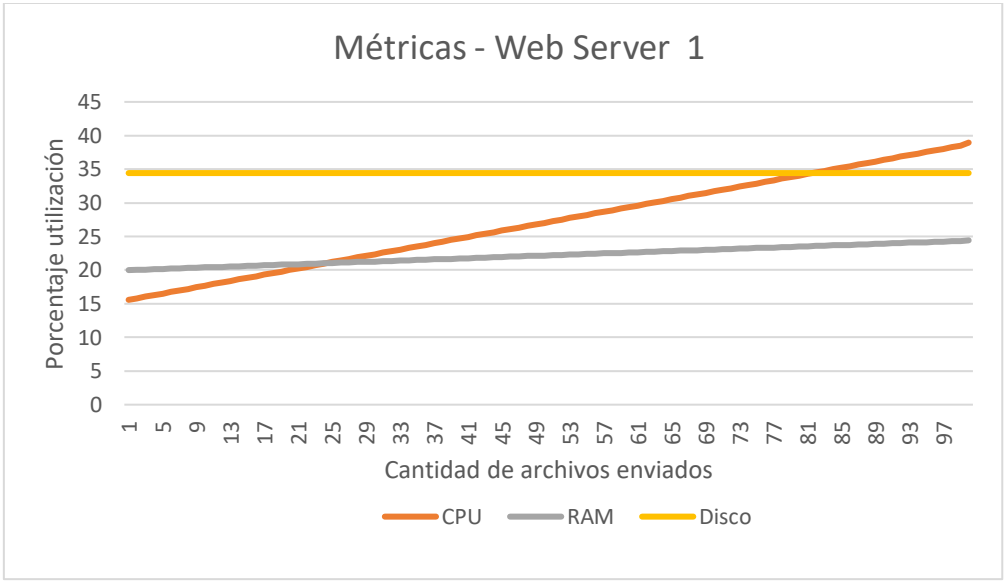

Figura 15. Métricas de los recursos de Web Server

La **Error! Reference source not found.** muestra el consumo de CPU, memoria y disco de la instancia del primer Web Server. Se observa que el consumo de CPU aumentó hasta llegar al 40%. El consumo de memoria se mantuvo por debajo de 25%.

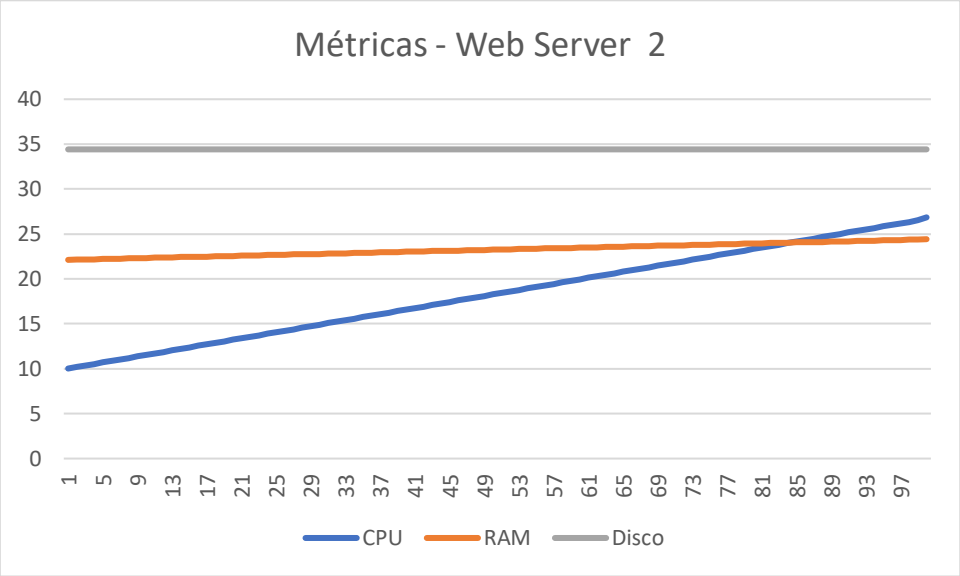

Figura 15. Métricas de los recursos de la segunda instancia del Web Server.

La **Error! Reference source not found.** muestra el consumo de CPU, memoria y disco de la instancia dedicada para el segundo Web Server. Se observa que el consumo de CPU aumentó hasta llegar por debajo del 30%. El consumo de memoria se mantuvo por debajo de 25%.

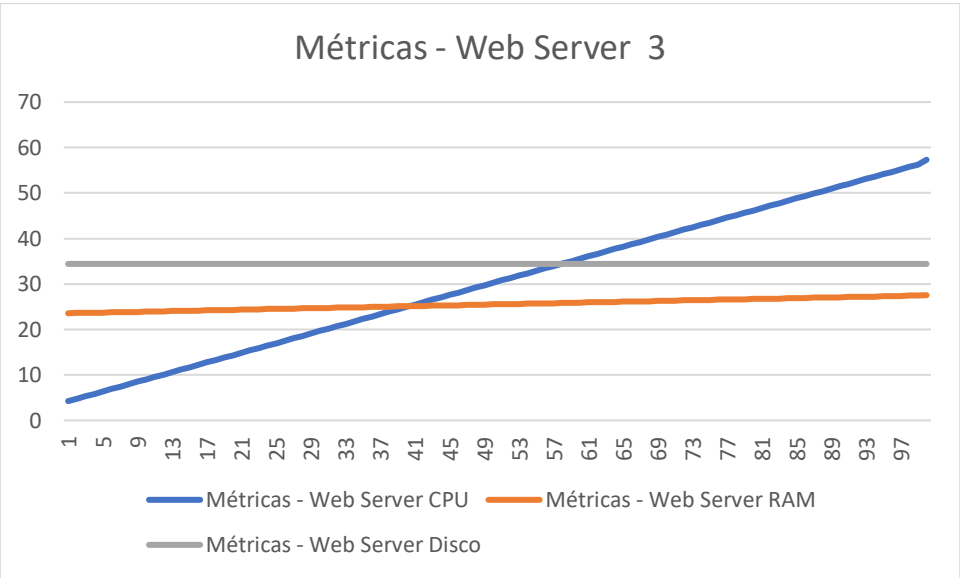

Figura 16. Métricas de los recursos de la tercera instancia del Web Server.

La Figura 16 muestra el consumo de CPU, memoria y disco de la instancia dedicada para el tercer Web Server. Se observa que el consumo de CPU aumentó hasta llegar por debajo del 60%. El consumo de memoria se mantuvo por debajo de 30%.

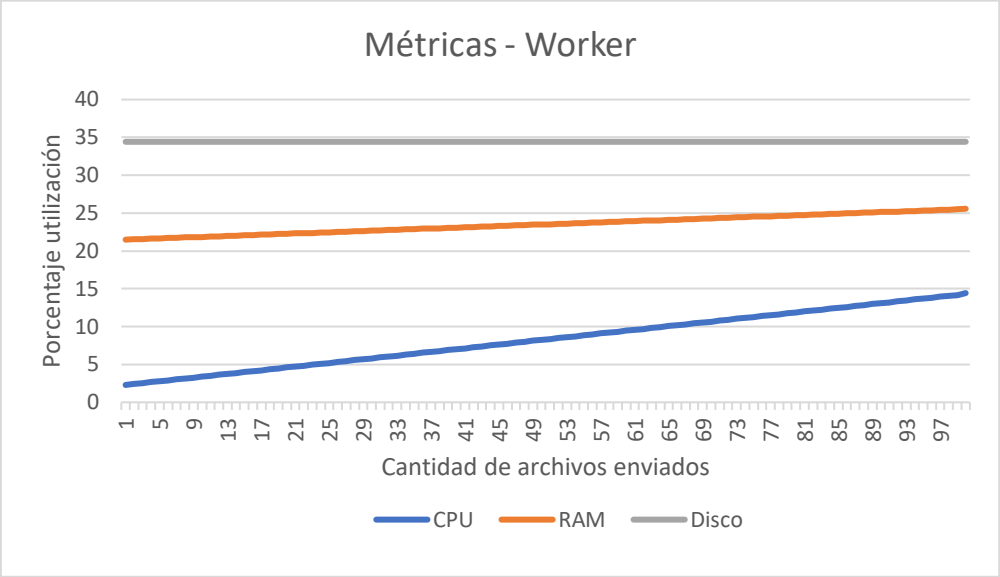

Figura 17. Métricas de los recursos de la tercera instancia del Web Server.

La Figura 17 muestra el consumo de CPU, memoria y disco de la instancia dedicada para el Worker. Se observa que el consumo de CPU aumentó hasta llegar por debajo del 15%. El consumo de memoria se mantuvo por debajo de 30%.

En general, se alcanzaron a procesar los 100 archivos y el consumo de memoria en los Web server fue menor comparado con la parte A del desarrollo.

9.1.2 Compresión Tar.gz2

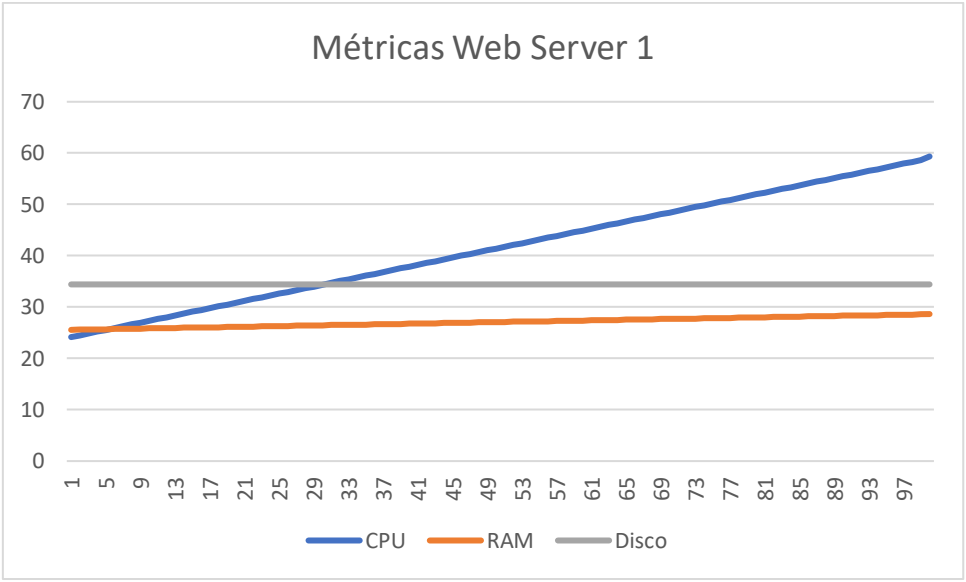

Figura 18. Métricas de los recursos de la primera instancia del Web Server.

La Figura 18 muestra el consumo de CPU, memoria y disco de la instancia dedicada para el primer Web Server. Se observa que el consumo de CPU aumentó hasta llegar por debajo del 60%. El consumo de memoria se mantuvo por debajo de 30%.

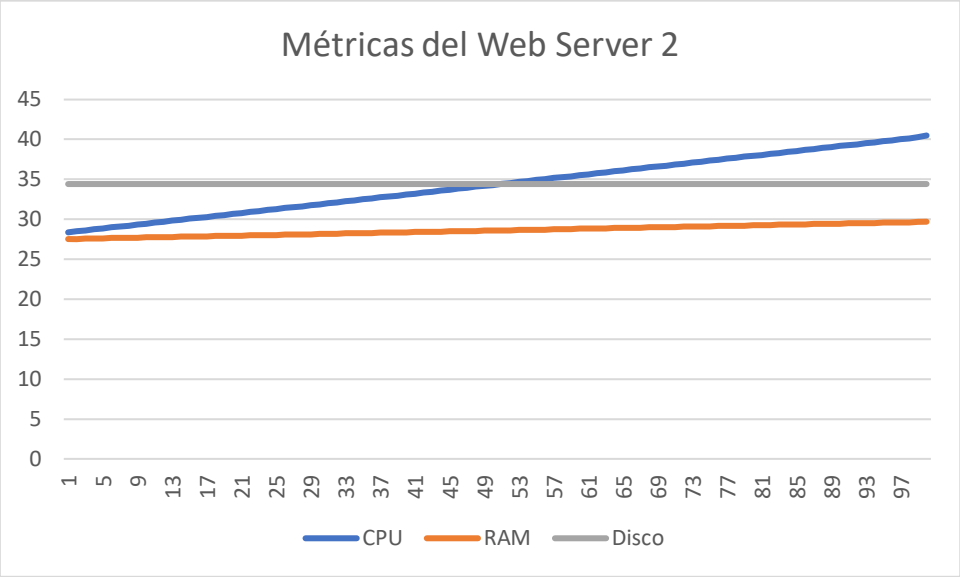

Figura 19. Métricas de los recursos de la segunda instancia del Web Server.

La Figura 19 muestra el consumo de CPU, memoria y disco de la instancia dedicada para el segundo Web Server. Se observa que el consumo de CPU aumentó poco comparado a la primera instancia, y se ubicó en 40%. El consumo de memoria se mantuvo por debajo de 30%.

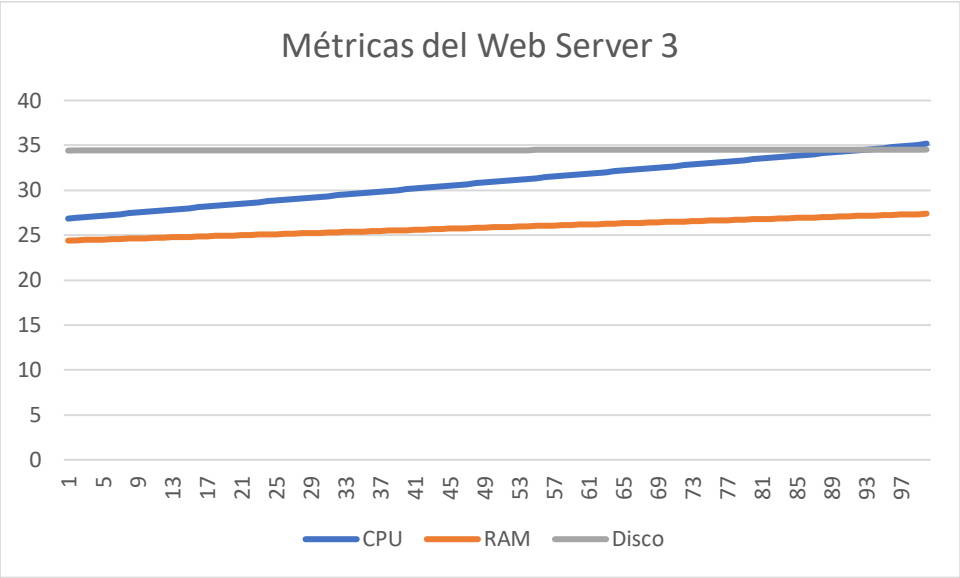

Figura 20. Métricas de los recursos de la tercera instancia del Web Server.

La Figura 20 muestra el consumo de CPU, memoria y disco de la instancia dedicada para la tercera instancia del Web Server. Se observa que el consumo de CPU aumentó poco comparado a la primera instancia, y se ubicó en 35%. El consumo de memoria se mantuvo por debajo de 30%.

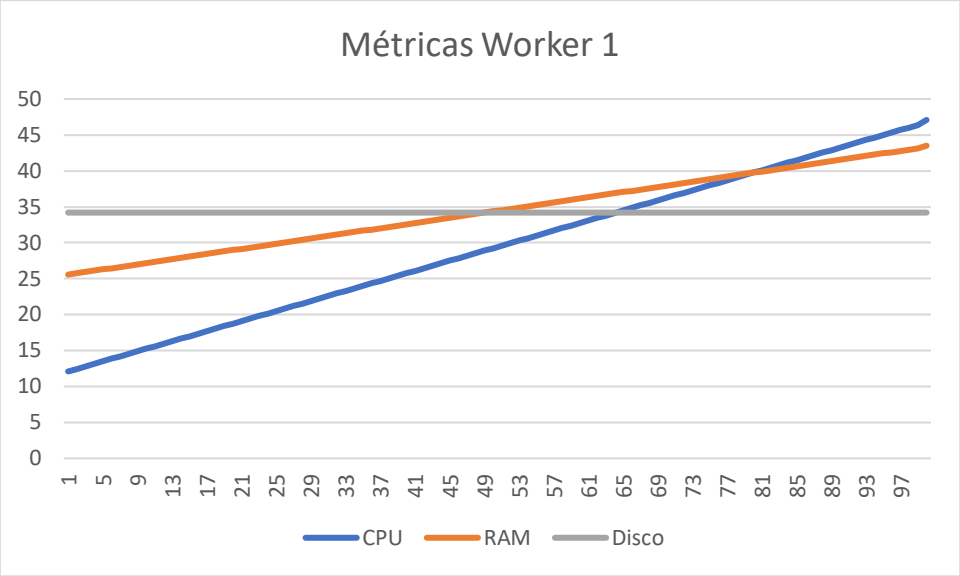

Figura 21. Métricas de los recursos de la primera instancia del Worker.

La Figura 21 muestra el consumo de CPU, memoria y disco de la instancia dedicada para el primer Worker. Se observa que el consumo de CPU aumentó poco comparado a la primera instancia, y se ubicó por debajo de 50%. El consumo de memoria se mantuvo por debajo de 45%.

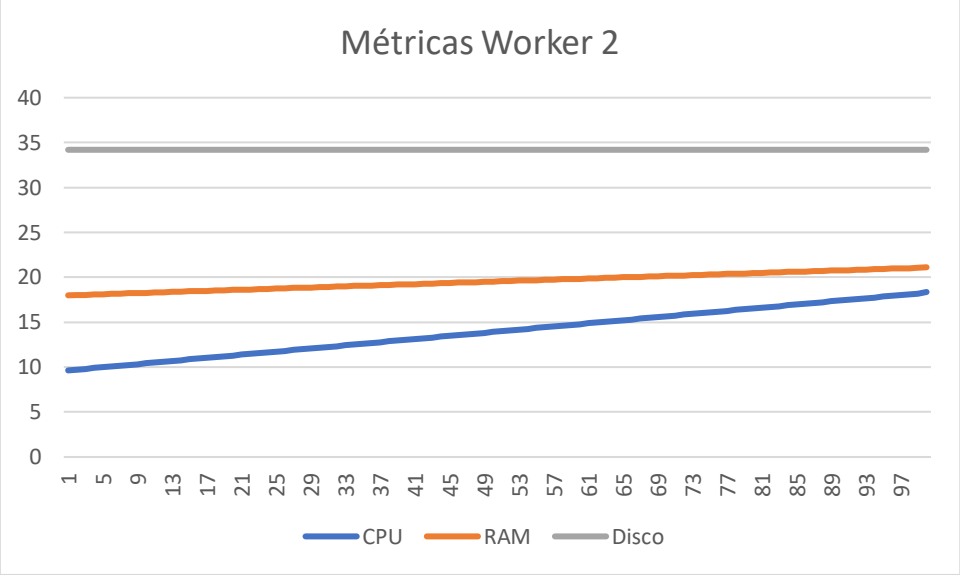

Figura 22. Métricas de los recursos de la segunda instancia del Worker.

La Figura 22 muestra el consumo de CPU, memoria y disco de la instancia dedicada para el segundo Worker. Se observa que el consumo de CPU aumentó poco comparado a la primera instancia, y se ubicó por debajo de 20%. El consumo de memoria se mantuvo cercano al 20%.

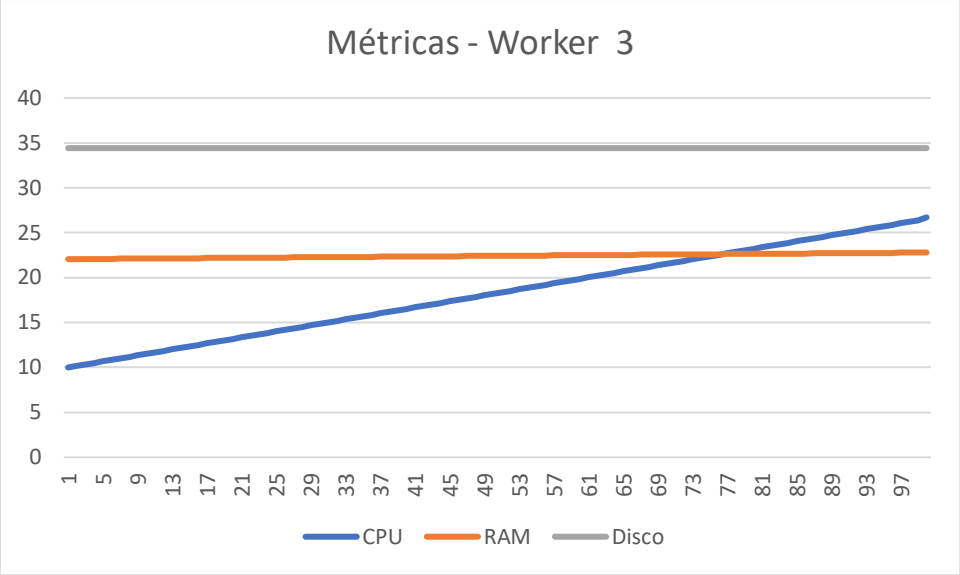

Figura 23. Métricas de los recursos de la tercera instancia del Worker.

La Figura 23 muestra el consumo de CPU, memoria y disco de la instancia dedicada para el tercer Worker. Se observa que el consumo de CPU aumentó poco comparado a la primera instancia, y se ubicó por debajo de 20%. El consumo de memoria se mantuvo por debajo del 25%.

En general, para la compresión Tar.bz2 se pudieron procesar los 100 archivos y el consumo de memoria y CPU de los web server fue menor comparada con la parte A del desarrollo.

9.2 Escenario 2

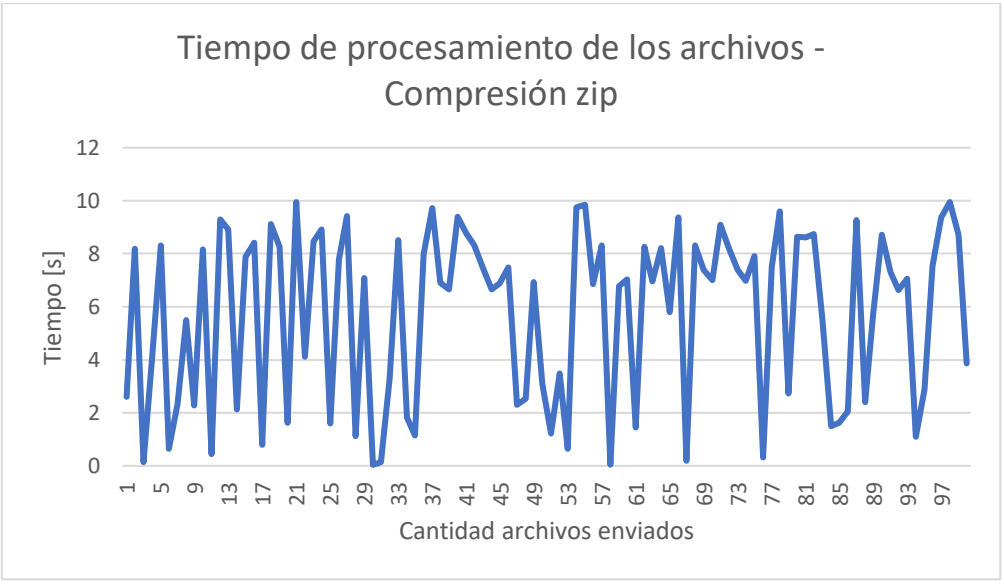

Figura 24. Tiempo de procesamiento de los archivos con compresión zip.

En la figura 24 se grafica el tiempo de procesamiento de los archivos con compresión zip. El tiempo medio es de 5.77 segundos. El tiempo máximo de procesamiento es de 9.95 segundos.

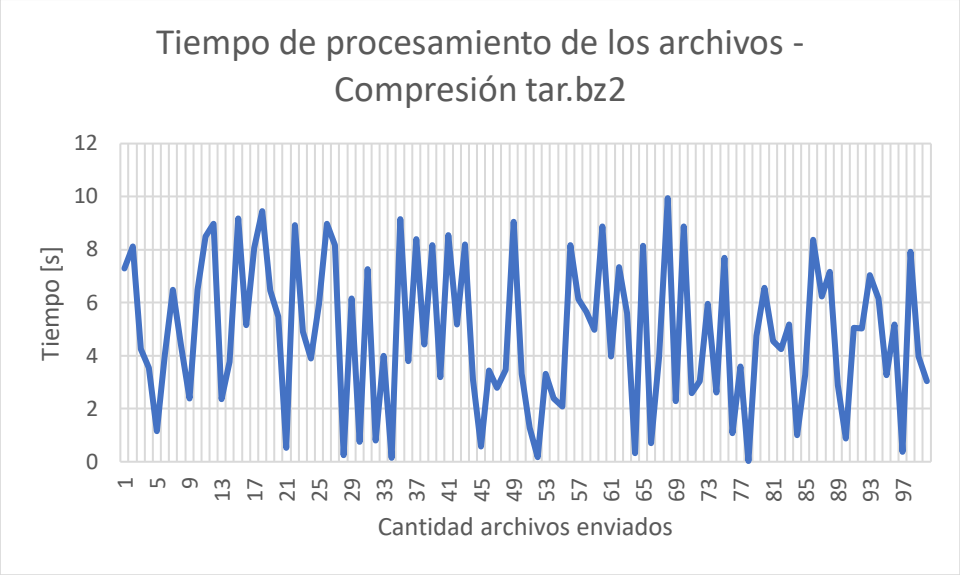

Figura 25. Tiempo de procesamiento de los archivos con compresión tar.bz2.

En la figura 25 se grafica el tiempo de procesamiento de los archivos con compresión tar.bz2. El tiempo medio es de 4.84 segundos. El tiempo máximo de procesamiento es de 9.46 segundos.

10. CONCLUSIONES

- Se adaptó la aplicación que permite la compresión de archivos a través de un sistema de encolamiento pub/sub 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. Asimismo, se cambió la instancia del file server por un bucket.
- 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.
- Se desplegó el grupo de instancias del servidor web en configuración multizona, lo que permite ser más tolerante a fallos del proveedor de nube.

ANEXOS

1. Metadatos *startup-script* web.

Clave*: startup-script*
Valor*:*
sudo apt-get update
sudo apt-get install -y wget
sudo apt install -y git

sudo apt-get update
sudo apt-get install ca-certificates curl gnupg
sudo install -m 0755 -d /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/debian/gpg | sudo gpg --yes --dearmor --dearmor -o /etc/apt/keyrings/docker.gpg
sudo chmod a+r /etc/apt/keyrings/docker.gpg
*echo *
*"deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/debian *
$(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt-get update
sudo apt-get -y install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
sudo curl -L "https://github.com/docker/compose/releases/download/1.29.2/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
sudo curl -L "https://github.com/docker/compose/releases/download/1.29.2/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
sudo chmod +x /usr/local/bin/docker-compose
git clone https://github.com/danip056/entrega_1.git
cd /entrega_1/
echo '{
##Encrypted
' > cloud_credentials.json
docker compose -f "docker-compose-web-prod.yml" up -d --build
curl -sSO https://dl.google.com/cloudagents/add-google-cloud-ops-agent-repo.sh sudo bash add-google-cloud-ops-agent-repo.sh --also-install

2. Metadatos *startup-script* worker.

Clave*: startup-script*
Valor*:*
sudo apt-get sudo apt-get update
sudo apt-get install -y wget
sudo apt install -y git
sudo apt-get update
sudo apt-get install ca-certificates curl gnupg
sudo install -m 0755 -d /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/debian/gpg | sudo gpg --yes --dearmor --dearmor -o /etc/apt/keyrings/docker.gpg
sudo chmod a+r /etc/apt/keyrings/docker.gpg
*echo *
*"deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/debian *
$(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt-get update

sudo apt-get -y install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin

sudo curl -L "https://github.com/docker/compose/releases/download/1.29.2/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
sudo curl -L "https://github.com/docker/compose/releases/download/1.29.2/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
sudo chmod +x /usr/local/bin/docker-compose
git clone https://github.com/danip056/entrega_1.git
cd /entrega_1/
echo '{
##Encrypted
}
' > cloud_credentials.json
docker compose -f "docker-compose-worker-prod.yml" up -d –build