Entrega 3 – Despliegue básico en la nube migración de una aplicación web a la nube pública

Paul A. Calvache Tapia, Fabian O. Ramírez, David A. Vásquez Pachón, Mateo Zapata Lopez
Desarrollo de Soluciones Cloud
Universidad de los Andes, Bogotá, Colombia
{pa.calvache, fo.ramirez50, da.vasquez11, m.zapatal2}@uniandes.edu.co
Fecha de presentación: abril 2 de 2023

Tabla de contenido

1 Consideraciones iniciales:

En un comienzo se despliega los componentes de la aplicación en GCP, de la siguiente manera.

- **Webserver:** está creado con una máquina virtual en Compute Engine, con la serie N1 – F1micro, con 1vCPU de 614 de RAM y 10 GB de almacenamiento, en la zona us-east 1b y autorizamos el HTTP traffic.
- **Worker:** está creado con una máquina virtual en compute engine, con la serie N1 – F1micro, con 1vCPU de 614 de RAM y 10 GB de almacenamiento, en la zona us-east 1b y autorizamos el HTTP traffic.
- **BDinstance:** está creado en una instancia Cloud SQL, con un motor de base de datos PostgresSQL, con una sola zona, con tipo de maquina liviano con 4 vCPU y 26 GB y tipo de disco SSD.

2 Arquitectura y descripción del sistema:

Sistema de compresión de archivos es una aplicación web accesible desde la red implementando las siguientes herramientas tecnológicas,

- Servidores web: 1 VM en GCP con SO Ubuntu
- Servidores worker: 1 VM en GCP con SO Ubuntu
- Servidor de archivos NFS: 1 VM en GCP con SO Ubuntu
- Instancia de Cloud SQL: con base de datos PostgresSQL
- Flask
- Python
- Celery y Redis
- Correo electrónico, Gmail.

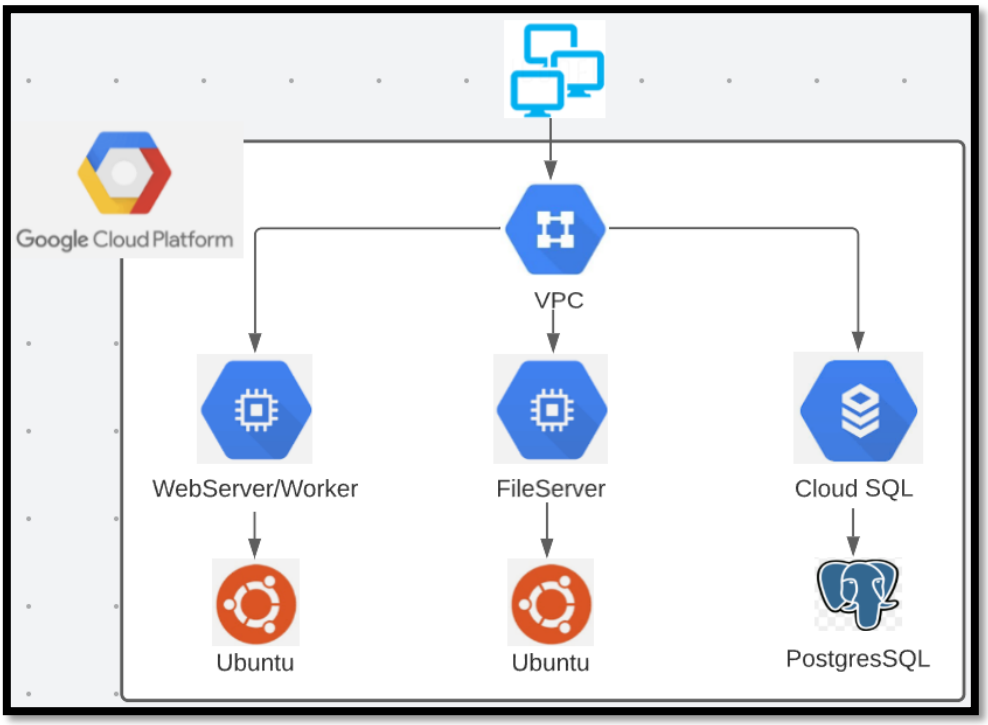

Imagen 1. Arquitectura Aplicación

3 Alcance del sistema:

Las funcionalidades para el programa de compresión de archivos son los siguientes:

Funcionalidad	Descripción
Sign-up	Creación de usuario, contraseña y correo de los nuevos usuarios
Login	Entrar a la aplicación con usuario y contraseña
Task	Vizualización de archivos y creación de nuevos trabajos de compresión
Upload	Subir los archivos para compresión

Tabla 1. Endpoints de la aplicación

4 Elementos tecnológicos del sistema:

Los elementos tecnológicos de la aplicación web, se explican en 3 partes:

1. Lógica: para la lógica de la página web se utilizó la librería Flask de Python para la creación del backend y el frontend, la librería celery y redis para el manejo de colas de trabajo (compresión de archivos) y postgres para el almacenamiento de datos.
   - Backend: Tiene varios componentes como el iniciador del programa, modelos de los datos y vistas que tiene todas las funciones de la página web. Hay un archvio “routes.py” donde se relacionan los endpoints (funciones y direcciones de la aplicación), que interactúan en el frontend.
     Esto nos sirve para crear una aplicación más flexible y escalable.

- Frontend: Esta sección se desarrolló mediante Javascript con lo cual se crearon la misma cantidad de endppints para la interacción usuario aplicación y poder trasmitir la información entre el frontend y el backend.
- Proceso de ejecución por colas: Este componente se encarga de manejar las colas de trabajo para la compresión de los archivos, Según el tiempo que se programe, pasa en batch todos los archivos para ser comprimidos y guardados, esto se realiza para todas las tareas en estado “uploaded”.

2. Diseño: este se realiza con Bootstrap y Javascript, con estos se realiza todas las vistas que tiene el usuario para interactuar con las funcionalidades de la aplicación.
3. Base de datos: La base de datos se desarrolló en PostgresSQL y se realiza el manejo de esta con PGAdmin.

Vista de arquitectura e información:

A continuación, se presentan los casos de uso de los usuarios al ingresar a la aplicación:

Casos:

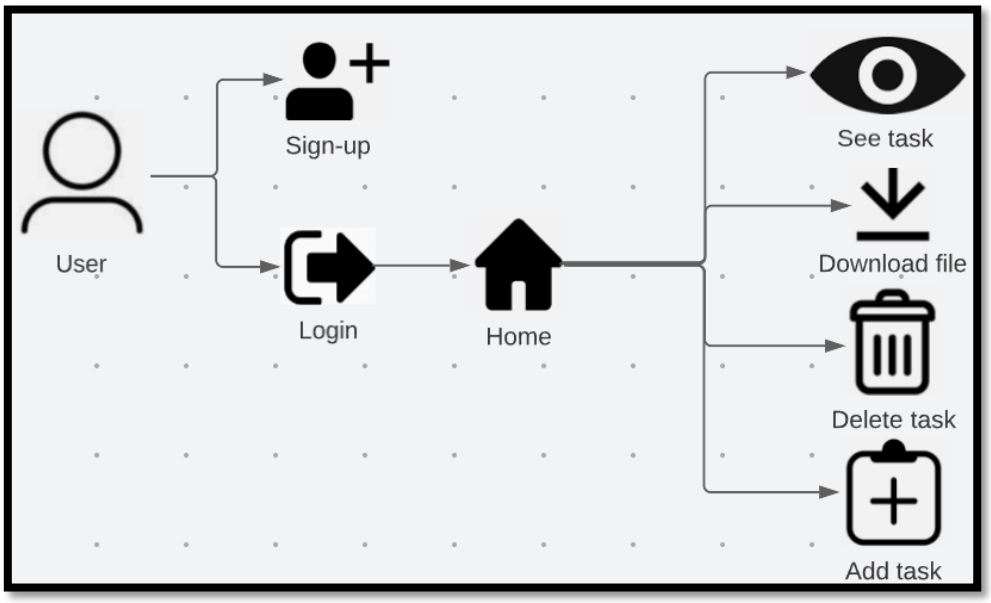

Imagen 2. Flujo de la aplicación para los usuarios.

Despliegue del modelo:

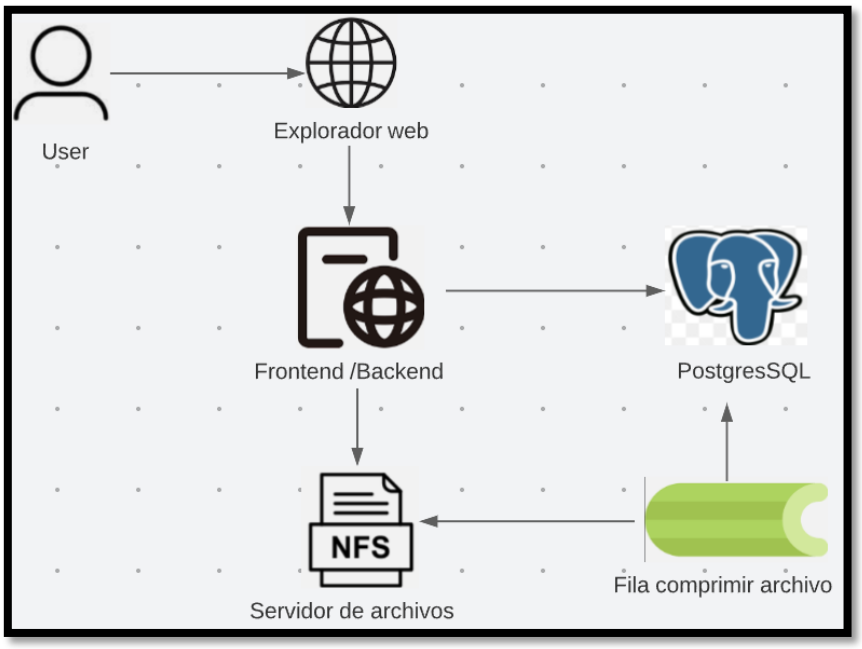

Imagen 3. Flujo de los componentes de la aplicación.

Vista de información:

La base de datos en PostgresSQL contiene 2 tablas, las cuales están relacionadas mediante llaves primarias y foráneas para vincular las tareas creadas por cada usuario, de esta manera solo se puede visualizar las tareas que cada usuario crea, brindando seguridad sobre las mismas:

- Usuario: Esta tabla está dedicada a la información del usuario, cuando se registra por primera vez (sign-up) se ingresan los siguientes datos a la tabla:
  - Usuario: El nombre del usuario que él escoge (llave primaria)
  - Email: Correo del usuario.
  - Password: La contraseña que el usuario coloca a su cuenta.
  - Task: Un número de identificación del usuario para crear tareas, este es autogenerado.

- Task: En esta tabla se guardan todos los datos de las tareas creadas por los usuarios:
  - Id_task: identificación de cada tarea creada
  - Filename: El nombre de cada archivo que se sube para comprimir
  - RutaArchivo: la ruta donde se guarda el archivo que se quiere comprimir
  - RutaCompresión: ruta donde se guarda el archivo ya comprimido
  - Status: pueden ser de dos tipos, upload y proccesed, upload cuando el archivo está recién creado y proccesed cuando ya se comprimió el archivo.
  - TipoConversión: el tipo de com presión que requiere el usuario para la tarea
  - FechaCarga: la fecha cuando se cargó el archivo
  - Usuario_task: aquí se tiene la llave de identificación del usuario (llave foránea)

Modelo entidad relación:

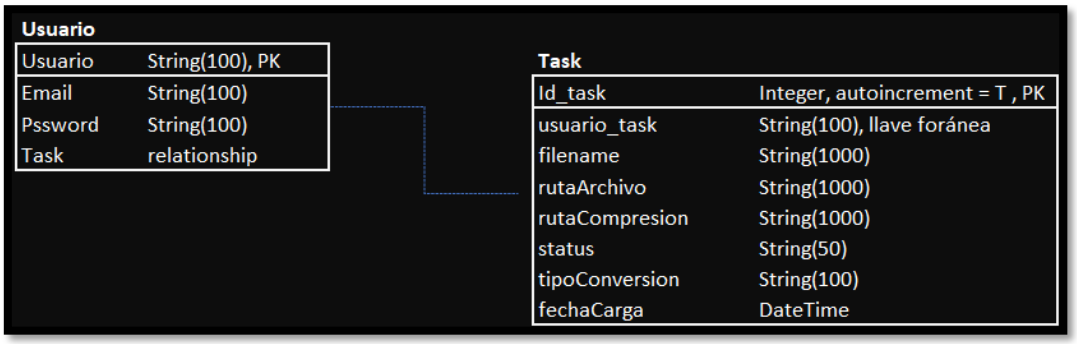

Imagen 4. Modelo entidad-relacion.

Ejecución del proceso de compresión de archivos

El worker está programado para obtener todas las tareas en estado “uploaded”, allí empezará a procesar cada tarea por colas, revisando el tipo de compresión y luego enviará un correo de confirmación al usuario registrado. La aplicación verifica y obtiene la ruta del campo “rutaArchivo” para encontrar el archivo que necesita comprimirse y envía en batch todos los archivos disponibles con la información del “tipoConversion” (formato de compresión solicitado al usuario), de allí hace la compresión de los archivos y los guarda en la “rutaCompresion”, para saber dónde se encuentra el archivo que acabó de comprimir.

Cuando se acaba de comprimir, al final del proceso corre la función para enviar las notificaciones de correo al e-mail del usuario registrado “email”.

Envío de correos:

Esta es la función que envía el correo cuando el archivo ya está comprimido, está atado a la ejecución de Celery, se utiliza Gmail y se da mensajes de consola para saber el estado de envío. Se toma la dirección de la base de datos del usuario (para esta entrega no se solicitó este servicio, por ende, fue deshabilitado).

```python
def enviarCorreo(email_to):
    # port number and server name
    smtp_port = 587                    #standard secure SMTP port
    smtp_server = "smtp.gmail.com"     #Google SMTP Server
    email_from = "" #Se debe crear cuenta en gmail para enviar correos, la anterior fue utilizada para enviar phising.
    pswd = ""
    # content of message
    message1 = "Por favor ingrese a nuestra plataforma para descargar su archivo."
    subject = "Sus archivos comprimidos estan listos"
    message = 'Subject: {}\n\n{}'.format(subject, message1)
    simple_email_context = ssl.create_default_context()
    try:
        print("Conectando al servidor...")
        TIE_server = smtplib.SMTP(smtp_server, smtp_port)
        TIE_server.starttls(context=simple_email_context)
        TIE_server.login(email_from, pswd)
        print("Conectado al servidor ...")
        print()
        print(f"Enviando notificacion a - {email_to}")
        TIE_server.sendmail(email_from, email_to, message)
        print(f"La notificacion fue enviada a - {email_to} ")
    except Exception as e:
        print(e)
    finally:
        TIE_server.quit()
    return "función enviarCorreo terminada"
```

Imagen 5. Función para envió de correo.

Sistema de archivos de red NFS: se montó en una VM independiente el servicio NFS para la aplicación, para esto se utilizó la librería NFS kernel-server, y se conectó con el webserver/ worker, también se tuvo que configurar las IP de clientes, para la libre comunicación entre las máquinas virtuales.

Subred: La subred se creó la aplicación Virtual Private Cloud (VPC) para la comunicación entre las VM y el Cloud SQL, se tuvieron que configurar las reglas del firewall y también las direcciones IP públicas para la comunicación entre las diferentes instancias, se usaron también la dirección de IP públicas para después realizar las pruebas de JMeter.

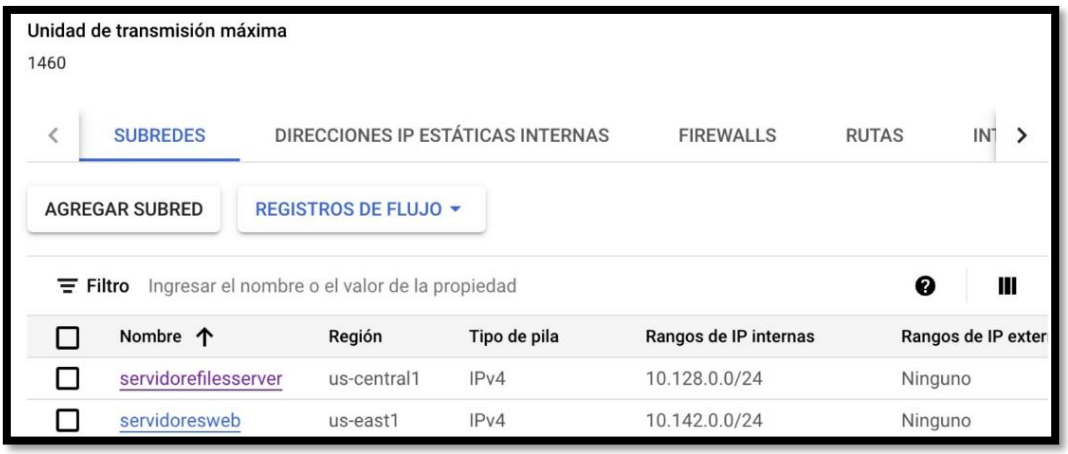

Imagen 6. Subredes.

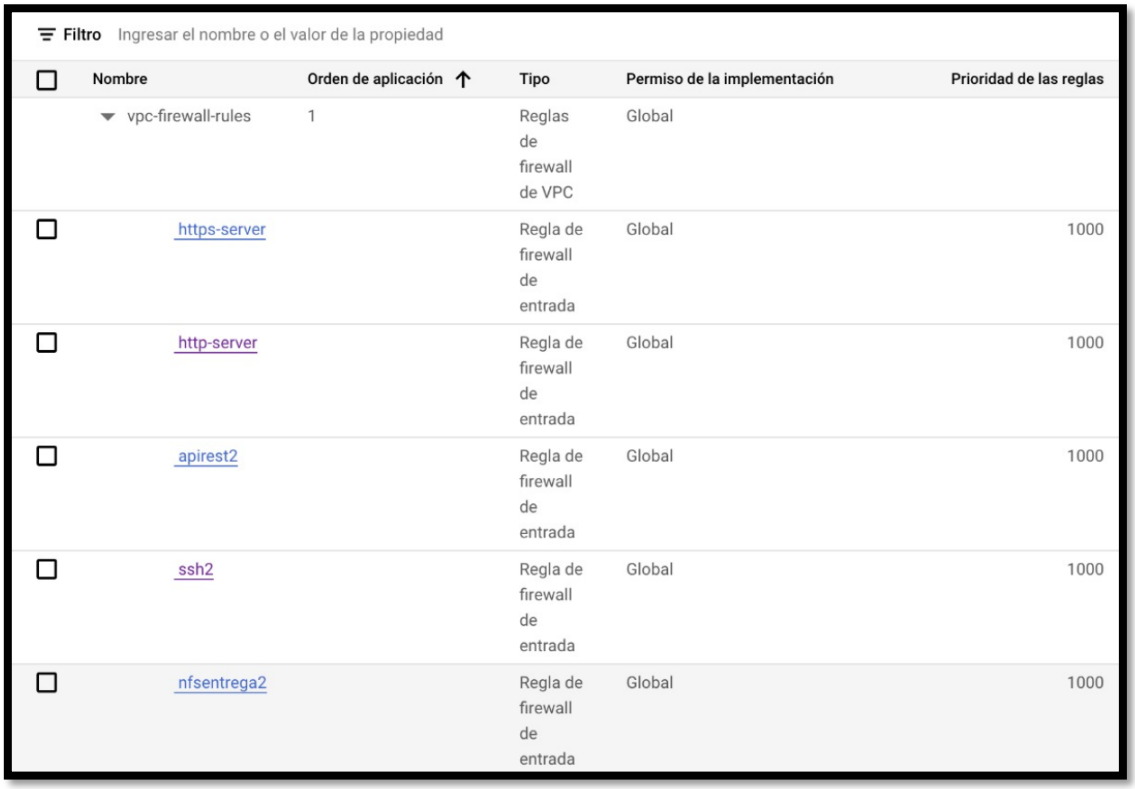

Imagen 7. firewall.

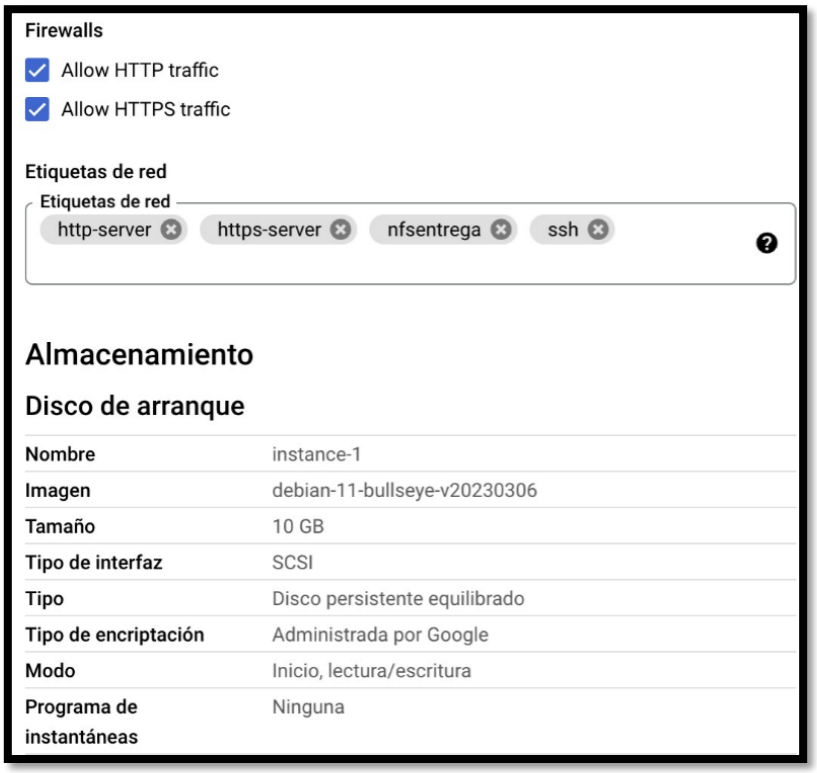

Imagen 8. Etiquetas de red.

5 Limitaciones:

- Limitación por la cantidad de correos: Dada que la cuenta creada es gratuita y utiliza SMTP tiene una limitación diaria de 500 correos que se pueden enviar y/o recibir, también se hace énfasis que esta función no fue requerida en la entrega, por ende, las credenciales fueron retiradas del código fuente.

- Limitación de tamaño de archivos: Ya que se realizó el despliegue con VM pequeñas con capacidad de 10 Gb, se realizó las pruebas de estrés con archivo de máximo 10 Mb y con una cantidad de 100 archivos.

- Limitaciones halladas en las pruebas de stress: Con referente a las pruebas de estrés realizadas en esta entrega notamos de que la CPU en uso alcanzo niveles del 50% con una prueba de 100 HTTP request en 10 segundos, lo que nos muestra que si aumentamos la cantidad de HTTP request puede llegar a colapsar la aplicación. También es válido aclarar que cuando se realizaron las pruebas con Jmeter la memoria llego a un 90% aproximadamente.

6 Conclusiones:

- Es importante configurar la red y subred para la correcta comunicación entre las instancias. (WebServer/worker, NFS y Cloud SQL). Las VPC que se escogieron para este trabajo, fueron regionales y las subredes fueron zonales, para ahorro de créditos y también por la naturaleza local del ejercicio.

- La configuración del NFS fue creada para compartir los archivos en un servidor para 2 rutas diferentes, una para alojar los archivos originales que sube el usuario, y la otra para los archivos procesados (comprimidos).
- GPC tiene el servicio Cloud SQL (Paas) que facilita el despliegue de la nube de bases de datos, ya que despliega un servicio exclusivo para motores de bases de datos.