Proyecto 1, Entrega 2 - Arquitectura, conclusiones y consideraciones

Integrantes:

- Daniel Santamaría Álvarez - 201720222
- Nestor Gonzalez – 201912670
- Álvaro Plata – 201820098
- Rafael Humberto Rodriguez Rodriguez - 202214371

Inicialmente, se crearon tres instancias en el servicio Compute Engine siguiendo las especificaciones detalladas en la guía. En cada instancia se desplegó un servicio diferente: una para desplegar la aplicación de Celery con redis (celery-instance), una para la aplicación de flask (flask-instance) y una para el manejo de archivos dentro de la red (nfs-instance).

Estado	Nombre	Zona	Recomendaciones	En uso por	IP interna	IP externa
✓	celery-instance	us-central1-a			10.128.0.3 (nic0)	35.206.64.148 (nic0)
✓	flask-instance	us-central1-a			10.128.0.2 (nic0)	35.209.141.223 (nic0)
✓	nfs-instance	us-central1-a			10.128.0.5 (nic0)	35.208.185.50 (nic0)

Además, se creó una instancia SQL para manejo de una base de datos PostgreSQL.

ID de instancia	Tipo	Dirección IP pública	Dirección IP privada	Nombre de la conexión con la instancia
✓ postgre-instance	PostgreSQL 14	34.68.80.135		swnube:us-central1:postgre-instance

Para migrar la aplicación a la nube, se hicieron múltiples cambios, especialmente para la conexión entre las máquinas virtuales.

Primero, se configuró que el folder /home en la instancia nfs será el punto de acceso para el manejo de los archivos de audio de la aplicación.

```
app.config["UPLOAD_FOLDER"] = "/nfs/home/files"
```

Además, debido a que cambiamos el motor de bases de datos de SQLite, de manera local, a PostgreSQL, se cambió la URI a la que SQLAlchemy se conecta para almacenar los datos de la aplicación.

```
app.config['SQLALCHEMY_DATABASE_URI'] = "postgresql://postgres:admin@34.68.80.135/postgres"
```

Ahora, para poder usar las tareas asíncronas de Celery, se creó una variable de entorno hacia la dirección IP de la instancia en la que se desplegó el servicio, que ya no es localhost.

```
celery_app = Celery(__name__, broker='redis://{}:6379/00'.format(os.environ.get("REDIS_INSTANCE_IP")))

@celery_app.task(name="start_conversion")
def start_conversion(*args):
```

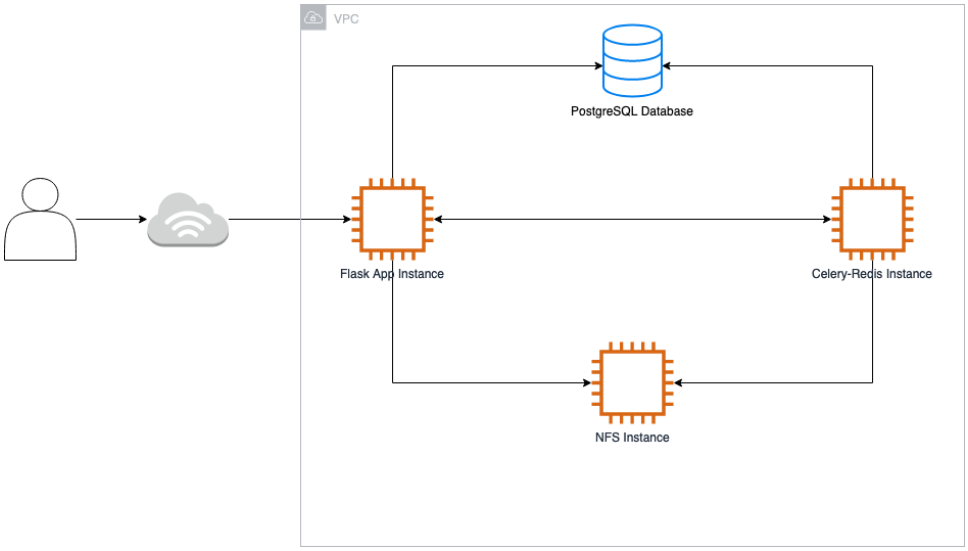
Y el llamado a las tareas de Celery ahora se realiza mediante la función apply_async que recibe los argumentos de la función y los delega a una cola, que en este caso se llama batch.

```
args = (task.id,in_route,out_route,in_ext,out_ext)
start_conversion.apply_async(args=args, queue="batch")
```

Finalmente, en el procesamiento en Celery, cuando ha finalizado la tarea, el servicio notifica a la aplicación de flask mediante un HTTP request. Para esto, también se definió una variable de entorno con la dirección IP de la instancia donde se ejecuta flask.

```
requests.post(f'http://{os.environ.get("API_INSTANCE_IP")}:5000/api/tasks/{task_id}/processed')
```

Arquitectura actual de la aplicación desplegada en Google Cloud:



La anterior imagen muestra la arquitectura actual de la aplicación ahora desplegada en Google Cloud. A continuación, se describen sus principales componentes y consideraciones:

1. Los usuarios acceden a la aplicación por medio de internet, a través de peticiones HTTP que llegan a la instancia en la cual se encuentra desplegada la aplicación (API) desarrollada en Flask.

2. La instancia “Flask Instance” tiene desplegada la API desarrollada. Una vez llegan las solicitudes de los usuarios se comunica con los demás componentes de la arquitectura. Por ejemplo, para solicitudes de Login se comunica con la base de datos PostgresQL para verificar las credenciales, para almacenar archivos se comunica con la instancia “NFS Instance” que cuenta con el sistema de archivos compartidos, y para crear tareas asíncronas se comunica con la instancia “Celery-Redis Instance”.

3. La base de datos “PostgresQL Database” cuenta con la información de las credenciales de los usuarios y de las tareas de conversión de archivos que se han solicitado y realizado. Esta base de datos fue creada con el servicio administrado SQL de Google Cloud, en lugar de desplegarla en una instancia de cómputo tradicional.

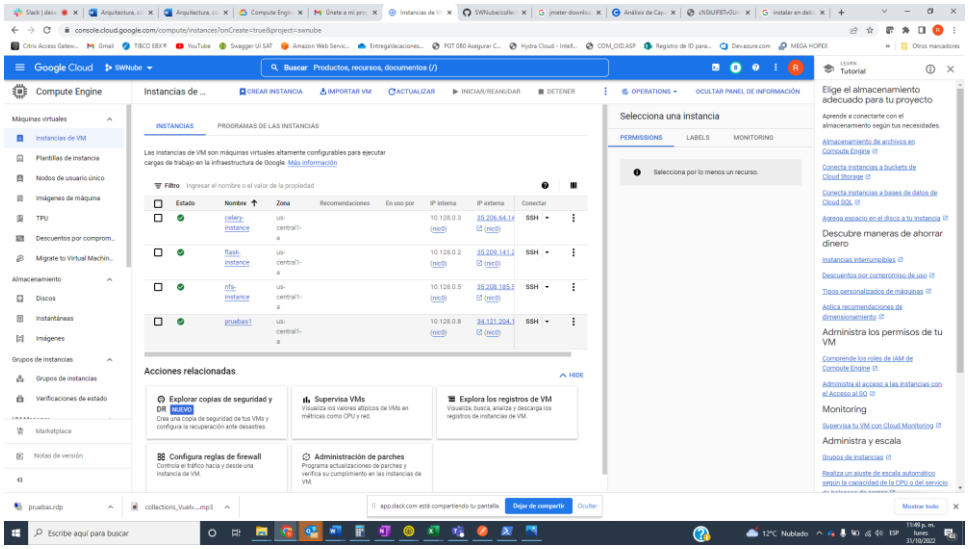
4. La instancia “NFS Instance” cuenta con los archivos que han subido los usuarios y sus conversiones a otros formatos. Esta instancia funciona como un Network File System para que los archivos sean visibles tanto a la API de Flask como para la aplicación de Celery.

5. La instancia “Celery-Redis Instance” cuenta con la aplicación de Celery y el servidor de Redis, que cumplirán la función de realizar tareas de conversión de formatos de archivos de forma asíncrona.

6. Una consideración importante es que se asignaron IPs estáticas a cada una de las instancias anteriores con el objetivo de simplificar el proceso de desarrollo y el análisis de capacidad. Sin embargo, para una posterior versión se tiene en cuenta que esto no sería necesario si se establece un grupo de auto escalamiento.

Escenarios de prueba.

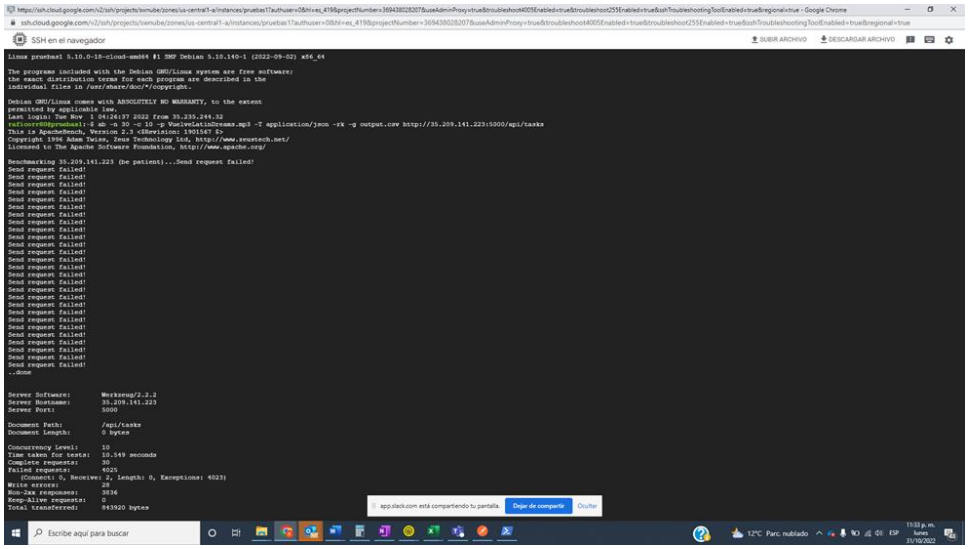
Se crea una instancia para pruebas en GCP con el nombre Pruebas 1

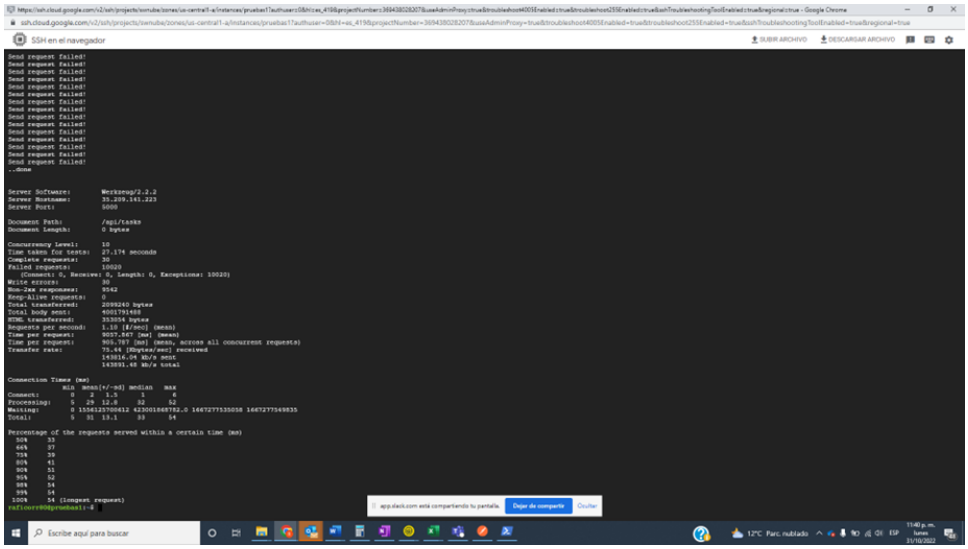


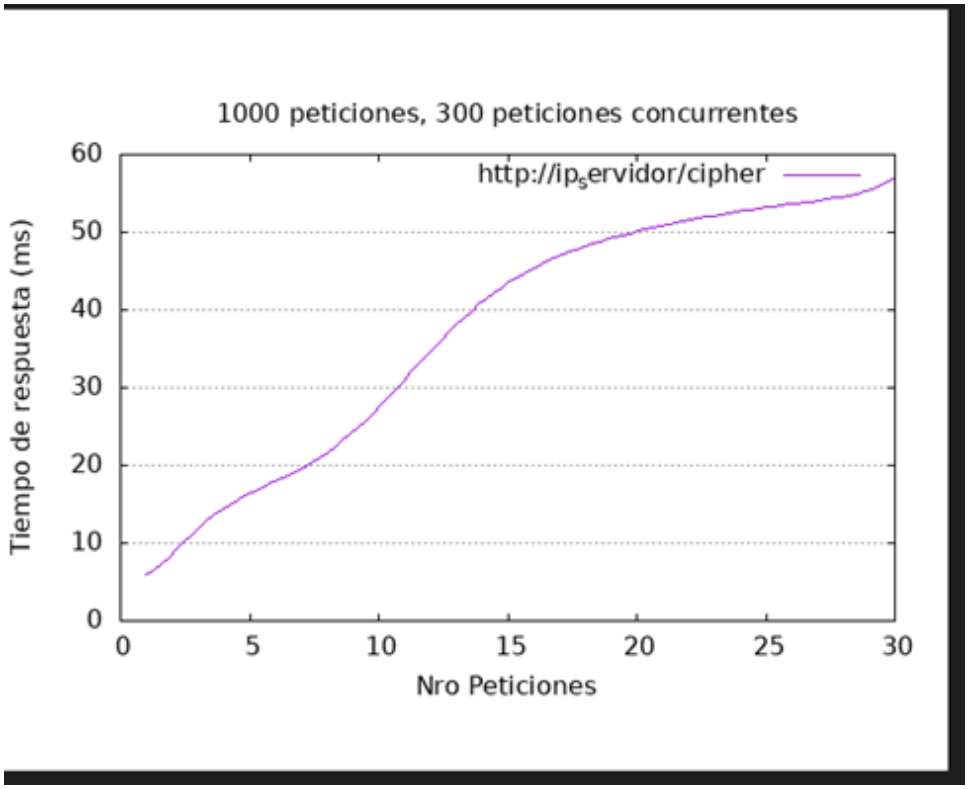
Se realiza las pruebas en apache Apache Benchmark

Escenario 1

Pruebas de estrés sobre el Modelo de Despliegue. Escenario 1.




SSH en el navegador
Send request failed!
Server Software: Werkzeug/2.2.2
Server Hostname: 35.239.141.223
Server Port: 5000
Document Path: /api/tasks
Document Length: 0 bytes
Concurrency Level: 10
Time taken for tests: 27.174 seconds
Complete requests: 30
Failed requests: 10020
(Connect: 0, Receive: 0, Length: 0, Exceptions: 10020)
Write errors: 30
Non-2xx responses: 9542
Keep-Alive requests: 0
Total transferred: 2099240 bytes
Total body sent: 4001791400
HTML transferred: 353054 bytes
Requests per second: 1.10 [#/sec] (mean)
Time per request: 9057.567 [ms] (mean)
Time per request: 905.757 [ms] (mean, across all concurrent requests)
Transfer rate: 75.44 [Kbyte/sec] received
143816.26 Kb/s sent
143891.69 Kb/s total
Connection Times (ms)
min mean[+/-sd] median max
Connect: 0 2 1.5 1 6
Processing: 5 29 12.8 32 62
Waiting: 0 1556125705612 423001868782.0 1667277535058 1667277549835
Total: 5 31 13.1 33 64
Percentage of the requests served within a certain time (ms)
50% 33
66% 37
75% 39
80% 41
90% 51
95% 52
98% 54
99% 54
100% 54 (longest request)
Escribe aquí para buscar


1000 peticiones, 300 peticiones concurrentes
http://ipservidor/cipher
Tiempo de respuesta (ms)
Nro Peticiones
0
10
20
30
40
50
60
0
5
10
15
20
25
30