Entrega 4 - Escenario y Pruebas de Estrés API REST y Batch

A. Plan de pruebas

- Parámetros de configuración:

Cada máquina VM de Compute Engine usada en la solución es del tipo N1 (f1-micro), con 1 vCPU, y 0.614 GiB de RAM más 10 GiB de almacenamiento. Se usa imagen de sistema operativo Ubuntu 18.04 LTS. Todas se ubican en el mismo segmento de red en un proyecto de Google Cloud. Ver Documento de Arquitectura en la wiki para mayor detalle a nivel de arquitectura.

Se usa bucket de Cloud Storage para almacenar todos los archivos de audio a procesar y procesados, con las sigui9entes características:

- Location type: “Region”, us-central1 (Iowa); reduce latencia a cambio de reducir disponibilidad y replicación
- Storage class: Standard
- Access control: Fine-grained, se marca Enforce public access prevention on this bucket

- Entorno de prueba:

Se usa un segundo bucket para guardar todos los archivos de configuración que requiere una VM API Rest para activarse. Mismas características del bucket de archivos:

- Location type: “Region”, us-central1 (Iowa); reduce latencia a cambio de reducir disponibilidad y replicación
- Storage class: Standard
- Access control: Fine-grained, se marca Enforce public access prevention on this bucket

Se arma una VM para lanzar las pruebas con Locust, para reducir sesgos derivados de un trayecto de red grande, al ubicar dicha VM en el mismo segmento de red de los componentes de la solución en nube. Esta máquina es del mismo tipo de las descritas anteriormente: N1 (f1-micro), 1 vCPU, 0.614 GiB RAM.

- Escenarios de prueba:

  Para la actividad se definen 2 escenario:

  1. Se desea probar cuál es la máxima, cantidad de requests HTTP por minuto que soporta la aplicación web con usuarios (al menos un usuario) que cuenten con 30 archivos disponibles.
  2. Se desea probar cuál es la máxima cantidad de archivos que pueden ser procesados por minuto en la aplicación local. Se usará una herramienta escrita en Python basada en Locust, ya que dio problema Apache Bench con requests tipo POST con archivo de audio.

Se usará una herramienta escrita en Python basada en Locust, dando continuidad a pruebas anteriores. Se dispara desde la instancia descrita en la sección Entorno de prueba.

- Criterios de aceptación:

  Los criterios de aceptación se definen según cada escenario de prueba

  1. El tiempo de respuesta promedio de la aplicación (carga de archivo) debe ser de máximo 1.500 ms, es decir 1.5s, y con porcentaje de errores menor a 1%. Si este tiempo no se

cumple, o si se generan más de 1% de errores en los requests disparados se concluye que el sistema NO soporta la cantidad de requests de la prueba.

2. Se lanzarán archivos de audio de peso mínimo de 5MB para cargar al sistema hasta llegar al punto de quiebre, que es que un archivo cargado se demore 10 minutos en ser tomado por el encolador.

B. Análisis de capacidad

El análisis de capacidad mediante pruebas de estrés se realizará mediante los 2 escenarios de pruebas mencionados anteriormente. Se ejecuta la prueba de estrés con el script de Python basado en Locust desde VM en cloud. Más información, ver repositorio de herramienta Locust:

https://github.com/lsolier/load-testing

- Escenario 1:

Archivo de configuración de prueba:

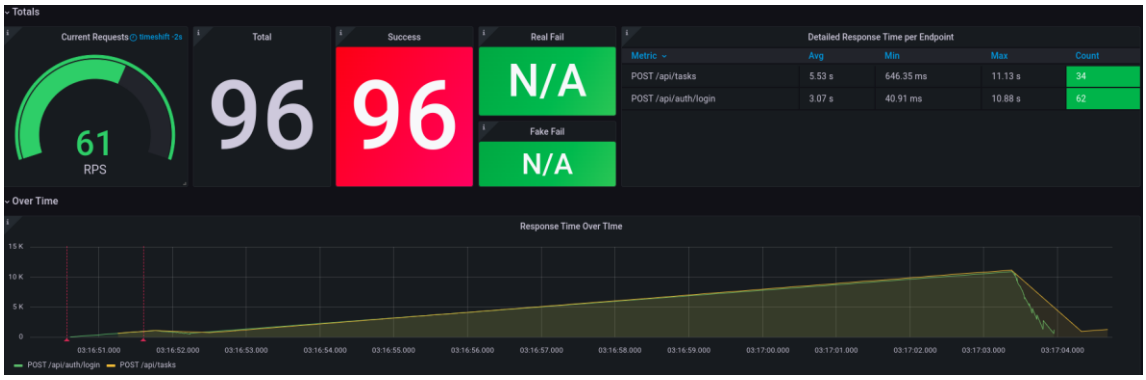
La prueba consiste en enviar 30 requests con una tase de nacimiento de requests de 30 por segundo; se ejecutará la prueba por un lapso de 1 minuto apuntando al IP PÚBLICO del load balancer.

```
(venv) luis_sajami@load-testing:~/load-testing$ cat locust.conf
locustfile = locustfile.py # file to run
headless = true # to run with the locust UI
host = http://35.209.134.18 # base url for the domains
users = 30 # amout of users that will run
master = false # to set master/worker state
spawn-rate = 30 # spawn rate per second
run-time = 1m # load testing  runtime
```

Se observa en la gráfica estabilidad al enviar las peticiones gradualmente hasta alcanzar 30 peticiones en alrededor 32s en donde cayó su comportamiento. La aplicación aun respondía con tiempos menores a 30s, despues de recibir 28 peticiones desde las 22:19:42 hasta las 22:20:10.

el endpoint **/api/auth/login** (autenticación ante plataforma para cargar archivos) tomó como máximo 21s

el endpoint **/api/tasks** (solicitud de carga de archivo) tomó un máximo de 22s



El api empezó a degradarse respondiendo en tiempo superior a los 16s a partir de las 20:20:10 con 50 request

- 18 request -> /api/tasks
- 32 request -> /api/auth/login

Conclusión

La capa web del sistema llega tener peticiones que no superar los 30sg por tiempo de respuesta.

La capa worker, escalo aumaticamente.

EL incremento de usuarios trae consigo tiempos de respuesta innecesarios.

La conclusión es que SI cumple con el criterio, aunque el comportamiento del Load Balancer genere confusión.

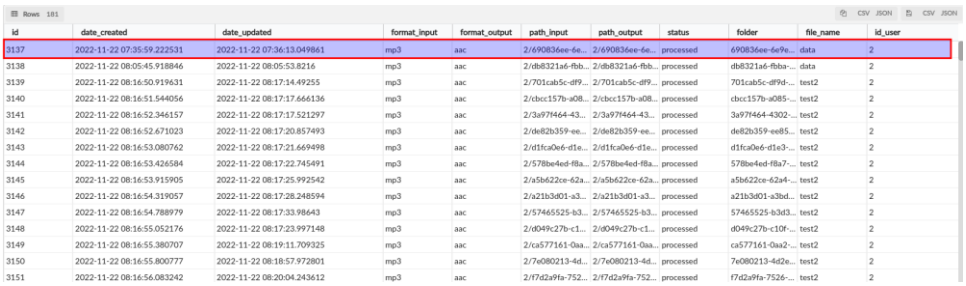
- Escenario 2:

Archivo de configuración de prueba:

La prueba consiste en enviar 30 requests con una tase de nacimiento de requests de 30 por segundo; se ejecutará la prueba por un lapso de 1 minuto apuntando al IP PÚBLICO del load balancer.

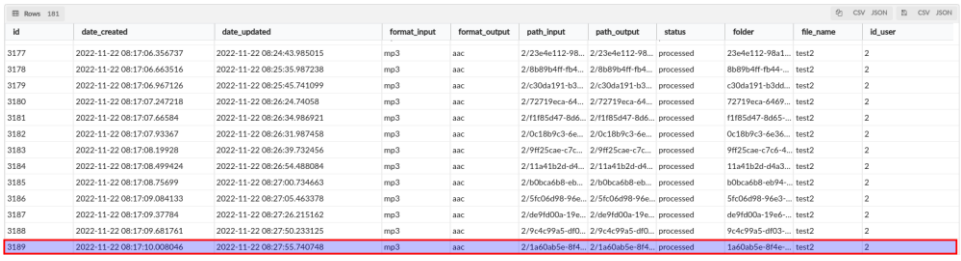
```
(venv) luis_sajami@load-testing:~/load-testing$ cat locust.conf
locustfile = locustfile.py # file to run
headless = true # to run with the locust UI
host = http://35.209.134.18 # base url for the domains
users = 30 # amout of users that will run
master = false # to set master/worker state
spawn-rate = 30 # spawn rate per second
run-time = 1m # load testing  runtime
```

Se observan las peticiones de conversión registradas en PostgreSQL y se ubica aquellas donde el delta de tiempo alcanza los 10 minutos; el primer archivo almacena fue a las **2022-11-22 07:35:59.222531** y el archivo que demoro más de 10 minutos en ser procesado fue almacenado a las **2022-11-22 07:36:13.049861**



id	date_created	date_updated	format_input	format_output	path_input	path_output	status	folder	file_name	id_user
3137	2022-11-22 07:35:59.222531	2022-11-22 07:36:13.049861	mp3	aac	2/690836ee-6e...	2/690836ee-6e...	processed	690836ee-6e9e-...	data	2
3138	2022-11-22 08:05:45.918846	2022-11-22 08:05:53.8216	mp3	aac	2/db8321a6-fbb...	2/db8321a6-fbb...	processed	db8321a6-fbba-...	data	2
3139	2022-11-22 08:16:50.919631	2022-11-22 08:17:14.49255	mp3	aac	2/701cab5c-df9...	2/701cab5c-df9...	processed	701cab5c-df9d-...	test2	2
3140	2022-11-22 08:16:51.544056	2022-11-22 08:17:17.666136	mp3	aac	2/cbcc157b-a08...	2/cbcc157b-a08...	processed	cbcc157b-a085-...	test2	2
3141	2022-11-22 08:16:52.346157	2022-11-22 08:17:17.521297	mp3	aac	2/3a97f464-43...	2/3a97f464-43...	processed	3a97f464-4302-...	test2	2
3142	2022-11-22 08:16:52.671023	2022-11-22 08:17:20.857493	mp3	aac	2/de82b359-ee...	2/de82b359-ee...	processed	de82b359-ee85-...	test2	2
3143	2022-11-22 08:16:53.080762	2022-11-22 08:17:21.669498	mp3	aac	2/d1fca0e6-d1e...	2/d1fca0e6-d1e...	processed	d1fca0e6-d1e3-...	test2	2
3144	2022-11-22 08:16:53.426584	2022-11-22 08:17:22.745491	mp3	aac	2/578be4ed-f8a...	2/578be4ed-f8a...	processed	578be4ed-f8a7-...	test2	2
3145	2022-11-22 08:16:53.915905	2022-11-22 08:17:25.992542	mp3	aac	2/a5b622ce-62a...	2/a5b622ce-62a...	processed	a5b622ce-62a4-...	test2	2
3146	2022-11-22 08:16:54.319057	2022-11-22 08:17:28.248594	mp3	aac	2/a21b3d01-a3...	2/a21b3d01-a3...	processed	a21b3d01-a3bd-...	test2	2
3147	2022-11-22 08:16:54.788979	2022-11-22 08:17:33.98643	mp3	aac	2/57465525-b3...	2/57465525-b3...	processed	57465525-b3d3-...	test2	2
3148	2022-11-22 08:16:55.052176	2022-11-22 08:17:23.997148	mp3	aac	2/d049c27b-c1...	2/d049c27b-c1...	processed	d049c27b-c10f-...	test2	2
3149	2022-11-22 08:16:55.380707	2022-11-22 08:19:11.709325	mp3	aac	2/ca577161-0aa...	2/ca577161-0aa...	processed	ca577161-0aa2-...	test2	2
3150	2022-11-22 08:16:55.800777	2022-11-22 08:18:57.972801	mp3	aac	2/7e080213-4d...	2/7e080213-4d...	processed	7e080213-4d2e-...	test2	2
3151	2022-11-22 08:16:56.083242	2022-11-22 08:20:04.243612	mp3	aac	2/f7d2a9fa-752...	2/f7d2a9fa-752...	processed	f7d2a9fa-7526-...	test2	2

Select general de tabla analizada



id	date_created	date_updated	format_input	format_output	path_input	path_output	status	folder	file_name	id_user
3177	2022-11-22 08:17:06.356737	2022-11-22 08:24:43.985015	mp3	aac	2/23e4e112-98...	2/23e4e112-98...	processed	23e4e112-98a1-...	test2	2
3178	2022-11-22 08:17:06.663516	2022-11-22 08:25:35.987238	mp3	aac	2/8b89b4ff-fb4...	2/8b89b4ff-fb4...	processed	8b89b4ff-fb44-...	test2	2
3179	2022-11-22 08:17:06.967126	2022-11-22 08:25:45.741099	mp3	aac	2/c30da191-b3...	2/c30da191-b3...	processed	c30da191-b3dd-...	test2	2
3180	2022-11-22 08:17:07.247218	2022-11-22 08:26:24.74058	mp3	aac	2/72719eca-64...	2/72719eca-64...	processed	72719eca-6469-...	test2	2
3181	2022-11-22 08:17:07.66584	2022-11-22 08:26:34.986921	mp3	aac	2/f1f85d47-8d6...	2/f1f85d47-8d6...	processed	f1f85d47-8d65-...	test2	2
3182	2022-11-22 08:17:07.93367	2022-11-22 08:26:31.987458	mp3	aac	2/0c18b9c3-6e...	2/0c18b9c3-6e...	processed	0c18b9c3-6e36-...	test2	2
3183	2022-11-22 08:17:08.19928	2022-11-22 08:26:39.732456	mp3	aac	2/9ff25cae-c7c...	2/9ff25cae-c7c...	processed	9ff25cae-c7c6-4...	test2	2
3184	2022-11-22 08:17:08.499424	2022-11-22 08:26:54.488084	mp3	aac	2/11a41b2d-d4...	2/11a41b2d-d4...	processed	11a41b2d-d4a3-...	test2	2
3185	2022-11-22 08:17:08.75699	2022-11-22 08:27:00.734663	mp3	aac	2/b0bca6b8-eb...	2/b0bca6b8-eb...	processed	b0bca6b8-eb94-...	test2	2
3186	2022-11-22 08:17:09.084133	2022-11-22 08:27:05.463378	mp3	aac	2/5fc06d98-96e...	2/5fc06d98-96e...	processed	5fc06d98-96e3-...	test2	2
3187	2022-11-22 08:17:09.37784	2022-11-22 08:27:26.215162	mp3	aac	2/de9fd00a-19e...	2/de9fd00a-19e...	processed	de9fd00a-19e6-...	test2	2
3188	2022-11-22 08:17:09.681761	2022-11-22 08:27:50.233125	mp3	aac	2/9c4c99a5-df0...	2/9c4c99a5-df0...	processed	9c4c99a5-df03-...	test2	2
3189	2022-11-22 08:17:10.008046	2022-11-22 08:27:55.740748	mp3	aac	2/1a60ab5e-8f4...	2/1a60ab5e-8f4...	processed	1a60ab5e-8f4e-...	test2	2

Threshold ubicado

Para calcular el threshold se usa la siguiente fórmula:

Procesamiento de conversión superior a 10min =

inicio request: 3137 - ; fin request: 3189

En el request número 54, el procesamiento de conversion tomo {"minutes":-10,"seconds":-45,"milliseconds":-732.702} es decir, aproximadamente 10 minutos (12 segundos)

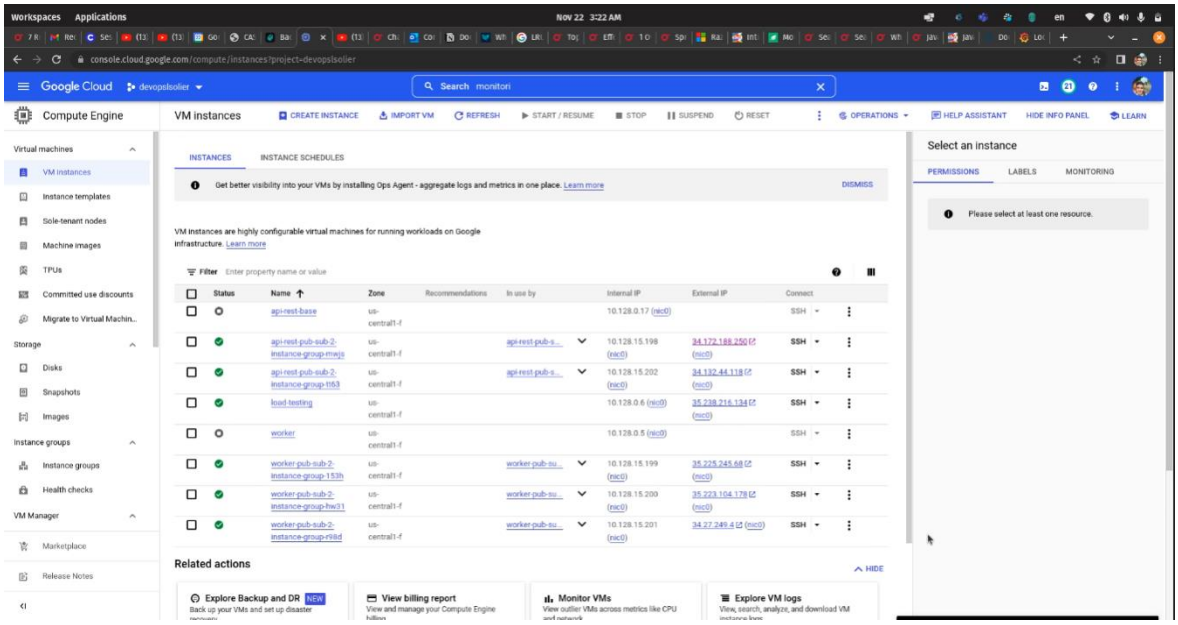
Conclusión:

Para el escenario de pruebas, se concluye que, a las 54 peticiones de conversión, procesar una nueva petición de conversión tomará mínimo 10 minutos.

- Información complementaria

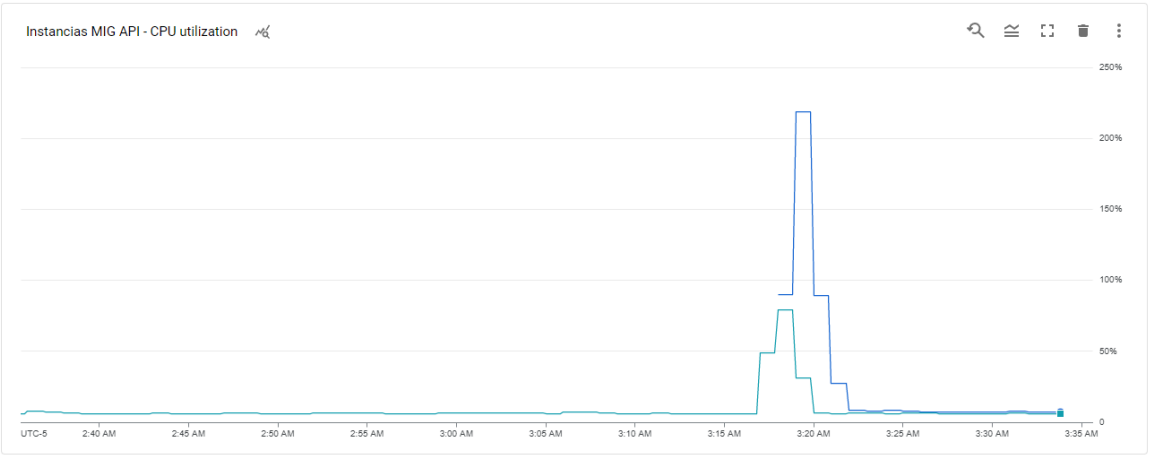
Está es data adicional capturada en la prueba del escenario 3, indicando distintas métricas como número de instancias arriba, carga sobre load balancer, y otros.

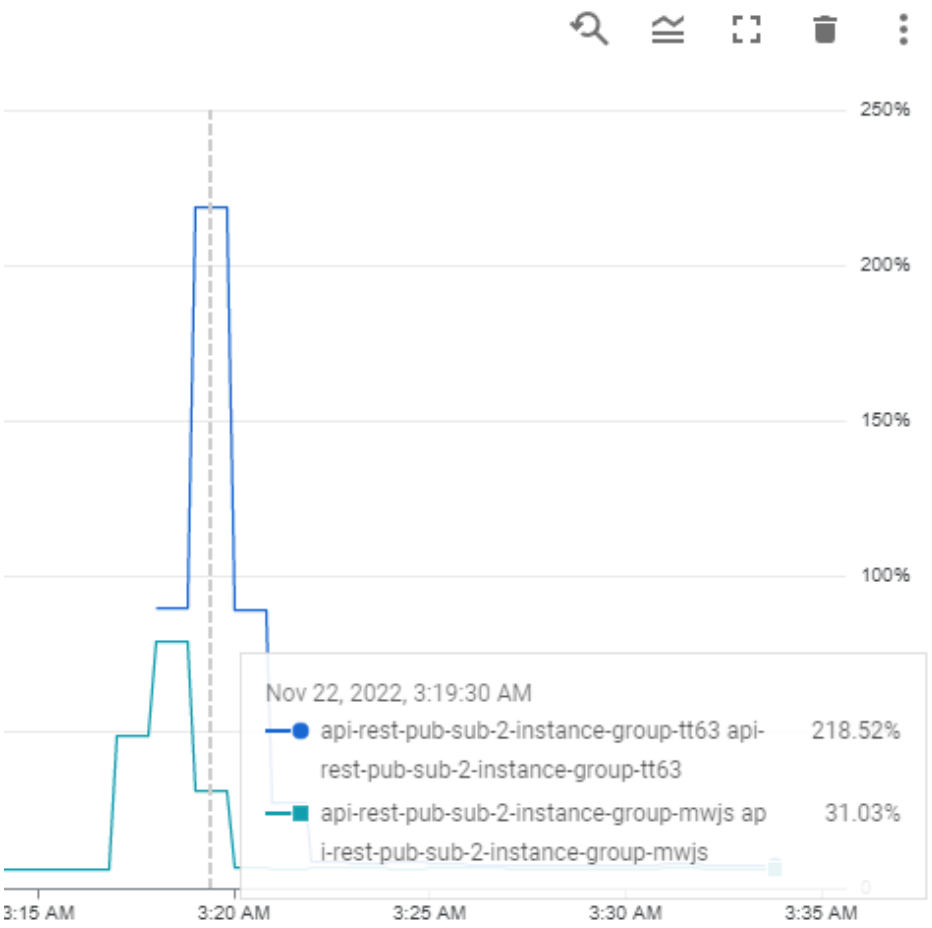
Instancias generadas por MIGs durante las pruebas de estrés



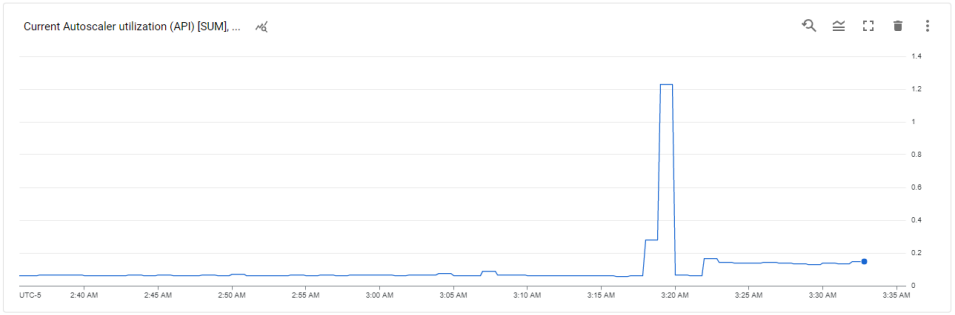
Número de instancias arriba por margen de hora:

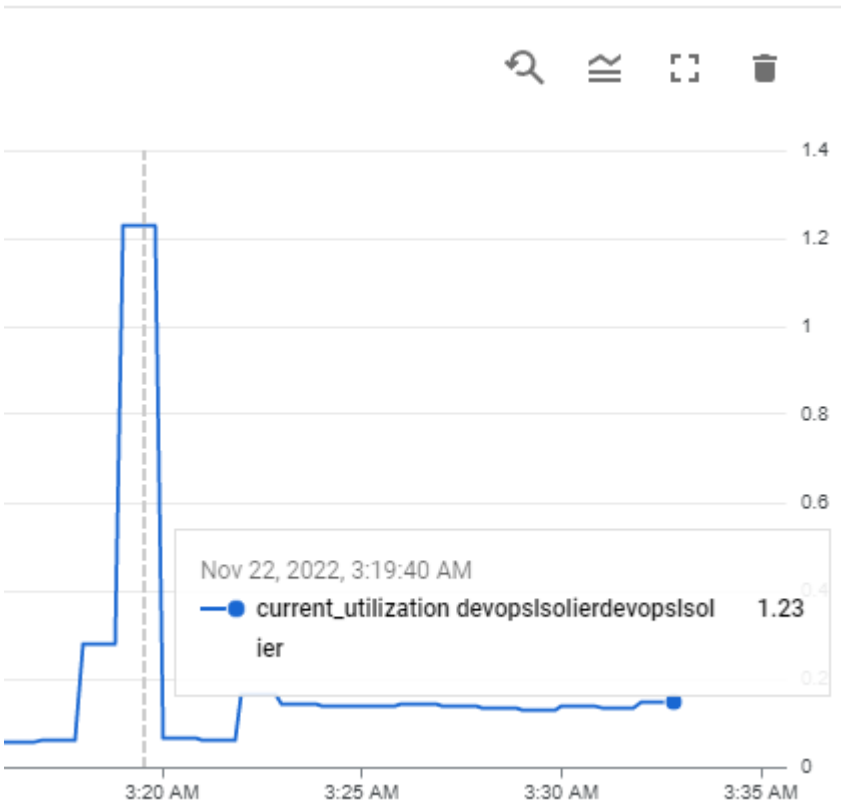
Consumo CPU por VM tipo API REST:



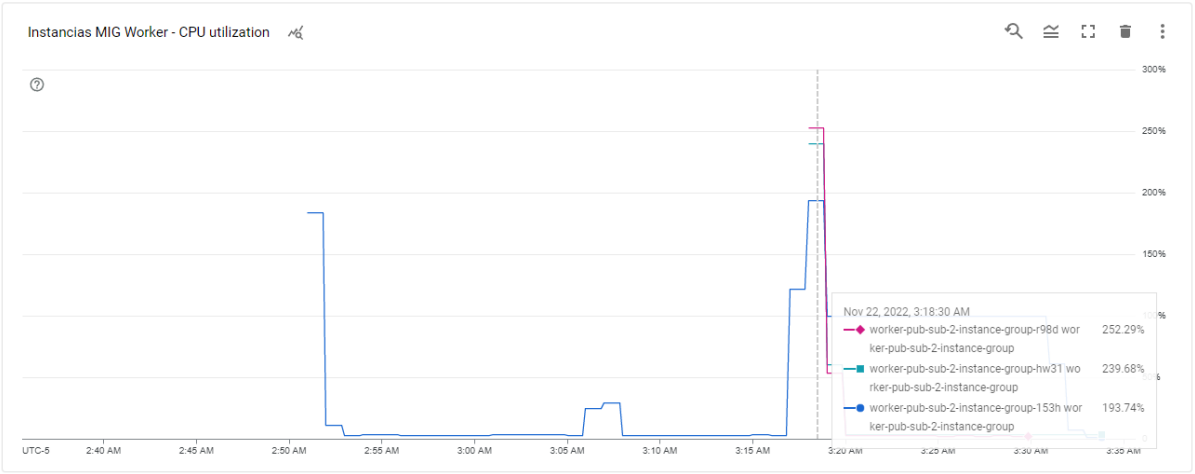


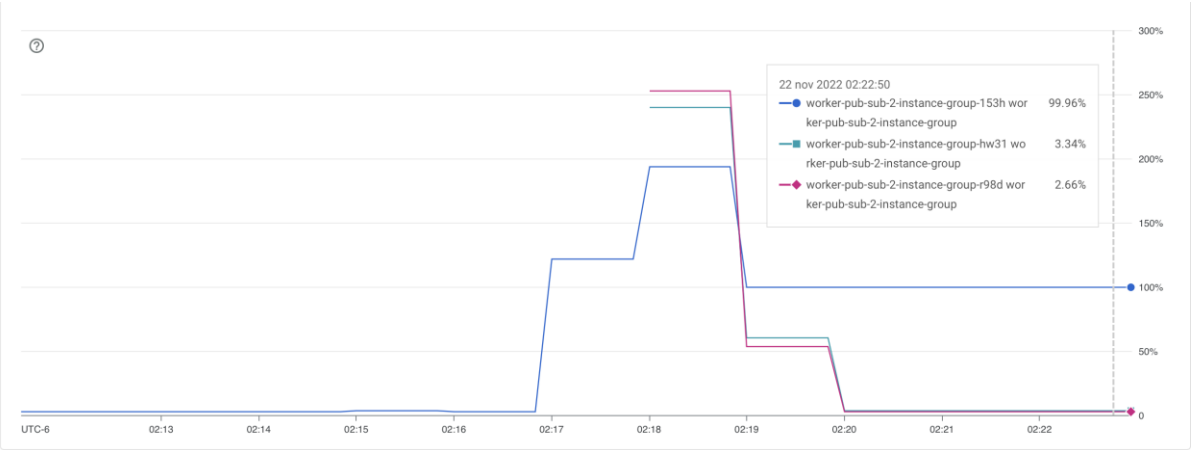
Comportamiento autoscaler API a lo largo del tiempo (número de instancias levantadas, y removidas)



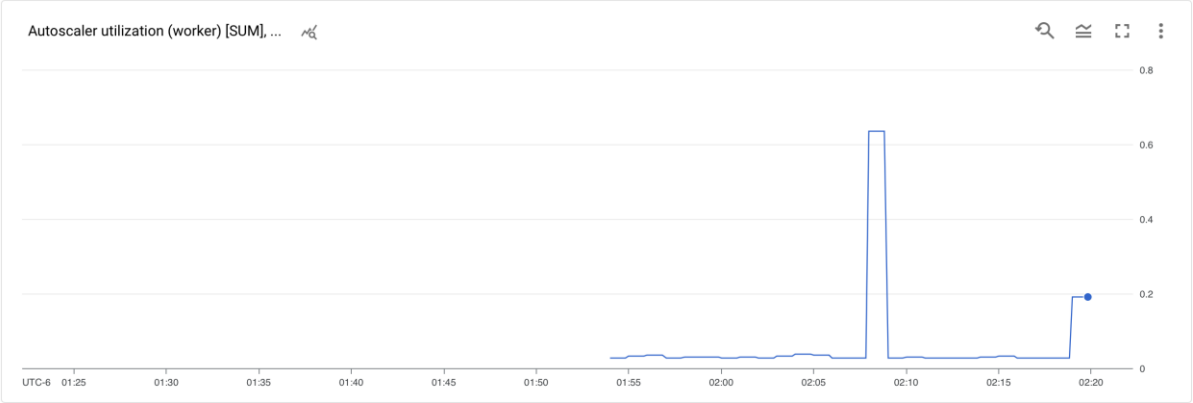


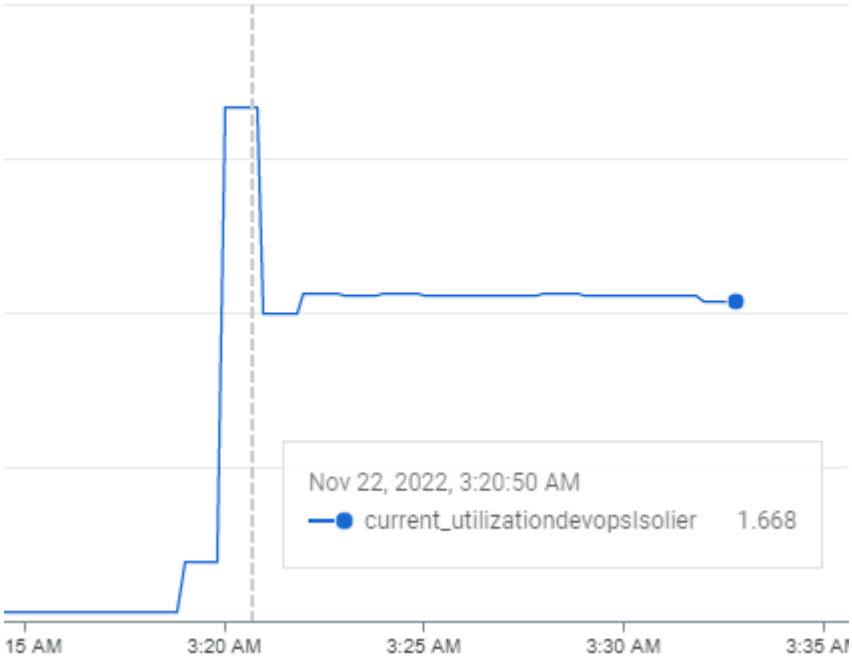
Consumo CPU por VM tipo Worker



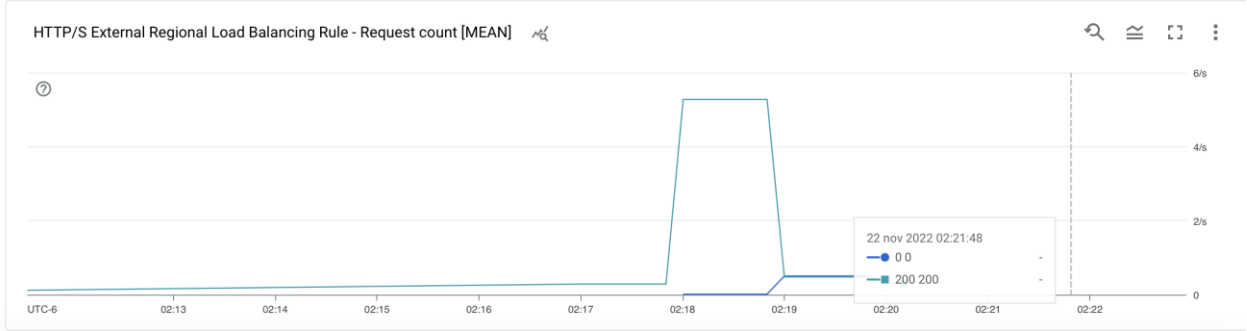


Comportamiento autoescaler Worker a lo largo del tiempo (*número de instancias levantadas, y removidas)*

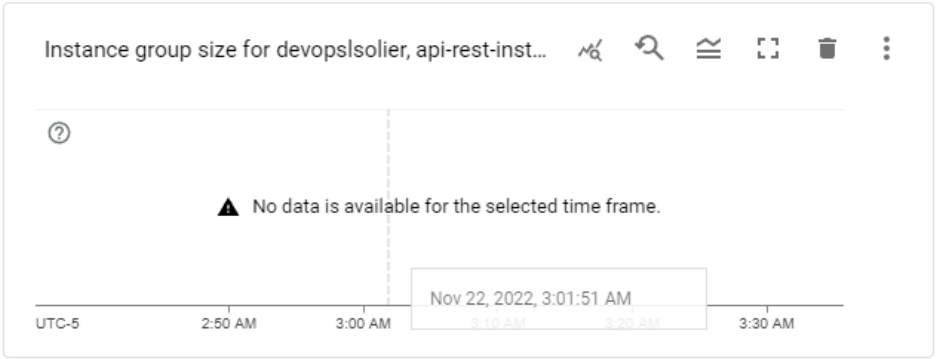




Número de peticiones enrutadas por load balancer



Monitoreo de tamaño de instancias de MIG de Worker parece estar bugeado:



Video de Pruebas:

https://uniandes-my.sharepoint.com/:v:/g/personal/e_melara_uniandes_edu_co/ESJbaU-wVdpImBhQ5-BYpokBtoz8ZblXMOCy-9xHUZkxIQ?e=7tFhVg