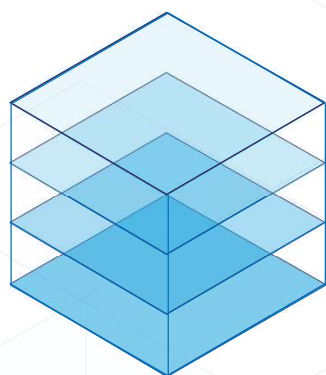




**MISO**  
Maestría en Ingeniería de Software

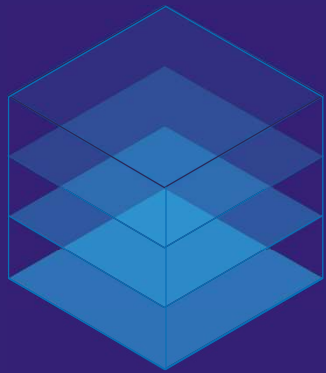


**MISO**

Maestría en Ingeniería de Software

# Experimentación

SaludTech – Alpes  
ByteBros



**MISO**

Maestría en Ingeniería de Software

## Experimento 1 - Escalabilidad con Concurrencia (Load Balancer)

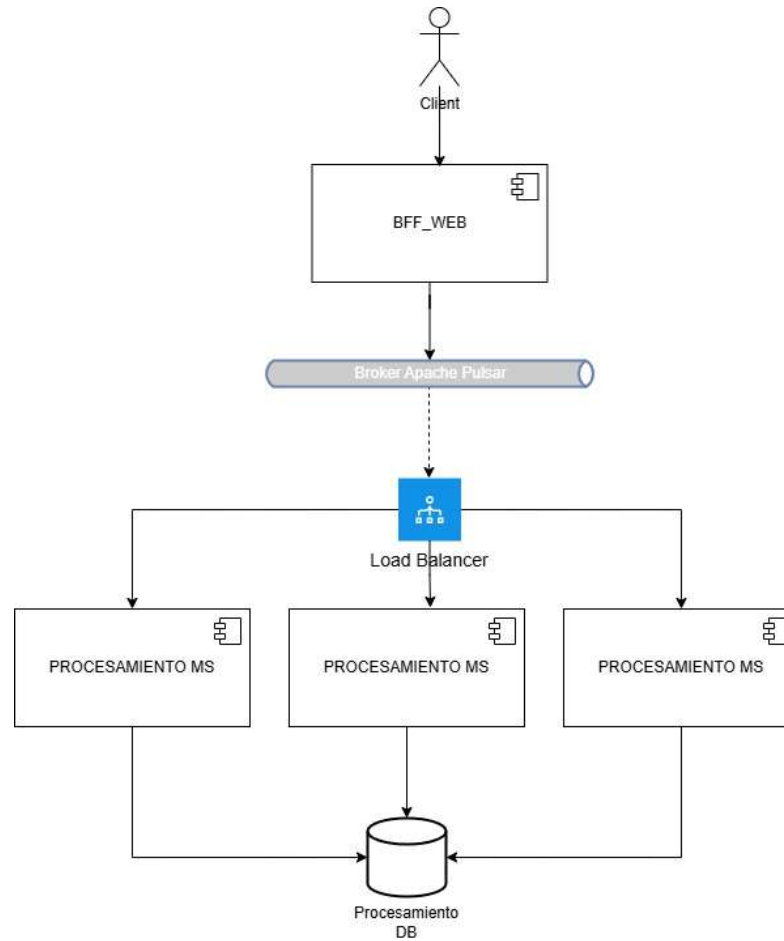
# Atributo de calidad 1: Escalabilidad

Escenario de calidad: Escalabilidad con Concurrencia (Load Balancer)			
Escenario #: 3	Evalúa la capacidad del sistema para manejar altos volúmenes de solicitudes concurrentes, distribuyéndolas de manera eficiente entre las instancias del sistema mediante un patrón de <i>Load Balancer</i> . El sistema debe ser capaz de procesar paquetes de imágenes, distribuyendo la carga entre diversas instancias de procesamiento, manteniendo la baja latencia, alta disponibilidad y tolerancia a fallos, especialmente al manejar eventos de procesamiento generados por el sistema.		
Fuente	El sistema que gestiona las solicitudes de procesamiento de las imágenes.		
Estímulo	El sistema recibe un paquete de imágenes que debe ser procesado. Este paquete es enviado como un evento para iniciar el procesamiento, disparando la asignación de tareas a las instancias disponibles.		
Ambiente	Operación normal con múltiples instancias activas, bajo una carga creciente de solicitudes. El sistema está preparado para gestionar picos de demanda y puede redirigir tareas entre instancias según la necesidad. En el caso de que una instancia falle, otra la sustituye sin interrumpir el flujo.		
Artefacto	El microservicio de procesamiento de imágenes		
Respuesta	El sistema distribuye las solicitudes de procesamiento de paquetes de imágenes entre varias instancias, asignando un paquete a cada instancia disponible. Si llegan más paquetes mientras una instancia está procesando, la carga es redistribuida entre las instancias restantes. El sistema debe asegurar que no haya cuellos de botella y que el procesamiento continúe sin interrupciones.		
Medida de la respuesta	El sistema debe procesar al menos el 99.9% de las solicitudes de imágenes sin superar un tiempo de respuesta de 2 segundos por paquete bajo carga máxima. El balanceador de carga debe ser capaz de distribuir la carga de manera eficiente sin que ninguna instancia supere el 75% de su capacidad.		
Decisiones Arquitecturales	Punto de sensibilidad	Tradeoff	Riesgo
Patrón de Load Balancer	Cuello de Botella en el Balanceo de Carga	Complejidad de Implementación vs. Escalabilidad	Falla en el Balanceador de Carga
Escalabilidad Dinámica	Sincronización de Instancias	Costos Operacionales vs. Redundancia	Sobrecarga de la Cola de Mensajes
Mensajería Asíncrona			
Justificación	El uso de un balanceador de carga es esencial para distribuir de manera eficiente las solicitudes entre las instancias, evitando cuellos de botella y mejorando la disponibilidad del sistema. La mensajería asíncrona permite distribuir tareas de forma flexible y permite que las instancias procesen solicitudes de manera independiente, manteniendo el rendimiento incluso en picos de carga. La escalabilidad dinámica permite ajustar el número de instancias automáticamente según la demanda, lo que optimiza los costos operacionales. Estas decisiones arquitecturales aseguran que el sistema maneje eficientemente altos volúmenes de solicitudes, mejorando la disponibilidad, reduciendo la latencia y respondiendo adecuadamente ante picos de demanda		

## Atributo de calidad 1: Escalabilidad

Escalabilidad con Concurrency (Load Balancer)

Diagrama de arquitectura



Hipótesis de diseño asociada al experimento	
Punto de sensibilidad	El balanceador de carga debe distribuir eficientemente las solicitudes entre las instancias disponibles, asegurando que ninguna supere el 75% de su capacidad y evitando cuellos de botella en el procesamiento. Además, debe redistribuir la carga si una instancia falla sin interrumpir el flujo de solicitudes.
Historia de arquitectura asociada	Como sistema de procesamiento de imágenes, cuando se reciben paquetes de imágenes para su procesamiento, dado que las instancias pueden enfrentar picos de demanda, quiero que el balanceador distribuya la carga dinámicamente, de manera eficiente y tolerante a fallos, para garantizar tiempos de respuesta óptimos y evitar sobrecargas en instancias individuales. Esto debe suceder en tiempo real y debe procesar al menos el 99.9% de las solicitudes de imágenes sin superar un tiempo de respuesta de 2 segundos por paquete.
Nivel de incertidumbre	El nivel de incertidumbre es medio, ya que el éxito del experimento depende en un 100% de la capacidad del balanceador de carga para gestionar eficientemente los picos de demanda, evitando latencias elevadas y cuellos de botella. Se reconoce que pueden ocurrir retrasos en la sincronización de instancias o en la redistribución de la carga en caso de fallas.

### Análisis de los resultados obtenidos

- 1- Indique si la hipótesis de diseño pudo ser confirmada o no
- 2- En caso de que la hipótesis se haya confirmado, explique las decisiones de arquitectura que favorecieron el resultado
- 3- En caso de que los resultados del experimento no hayan sido favorables, explique por qué y cuáles cambios realizaría en el diseño

1. El sistema es capaz de distribuir las solicitudes entre las instancias disponibles, sin que ninguna supere el 75% de su capacidad.
2. La hipótesis se cumplió gracias a la aplicación del patrón del load balancer, la escalabilidad dinámica, la arquitectura utilizando mensajería asíncrona por medio de apache pulsar entre el bff y el ms de procesamiento, además de la táctica de Monitor la cuál le permitieron al sistema identificar con rapidez las instancias que presentaban errores y retirarlas de la operación del sistema. Debido al desacoplamiento del microservicio de procesamiento pudo escalar de manera independiente, escuchando los eventos del bff para que cuando se disparara un comando, ejecutara el evento.

Se obtuvieron los siguientes resultados:

Buen rendimiento: Tiempo de respuesta promedio de 228 ms, lo cual es rápido (menos de 2 segundos).

Baja latencia mínima (90 ms), lo que indica que el servidor puede manejar respuestas rápidas en algunos casos.

Sin errores (0.00%), lo que significa que todas las solicitudes fueron exitosas.

Tiempo máximo de 1,350 ms: Aunque el promedio es bajo, algunas solicitudes tardaron más de 1 segundo. Puede ser una señal de sobrecarga en momentos puntuales.

Desviación estándar de 179.42 ms, lo que indica cierta variabilidad en los tiempos de respuesta.

Ninguna instancia supera el 75% de uso, lo máximo que alcanza una instancia es el 49% de uso de CPU.

## Evidencias

### 1- Presente evidencias de los resultados obtenidos en el experimento.

Se configura el JMeter:

Propiedades del Hilo:

Número de Hilos: 1000

→ Simula 1000 usuarios virtuales concurrentes en la prueba.

Periodo de Subida (Ramp-Up Period) en segundos: 60

→ Los 1000 hilos se iniciarán en 60 segundos, es decir, JMeter agregará aproximadamente 16-17 usuarios por segundo hasta alcanzar los 1000.

Contador del Bucle: 4

→ Cada hilo ejecutará 4 iteraciones antes de finalizar. Si estuviera en "Sin fin", se ejecutaría de manera indefinida.

Posible impacto:

Esta configuración genera una carga muy alta en un tiempo relativamente corto (60s). Si el sistema no está bien preparado, podría colapsar debido al número de solicitudes simultáneas.

Después se configura el JMeter con la dirección del balanceador de carga:

<http://34.54.118.91:80>

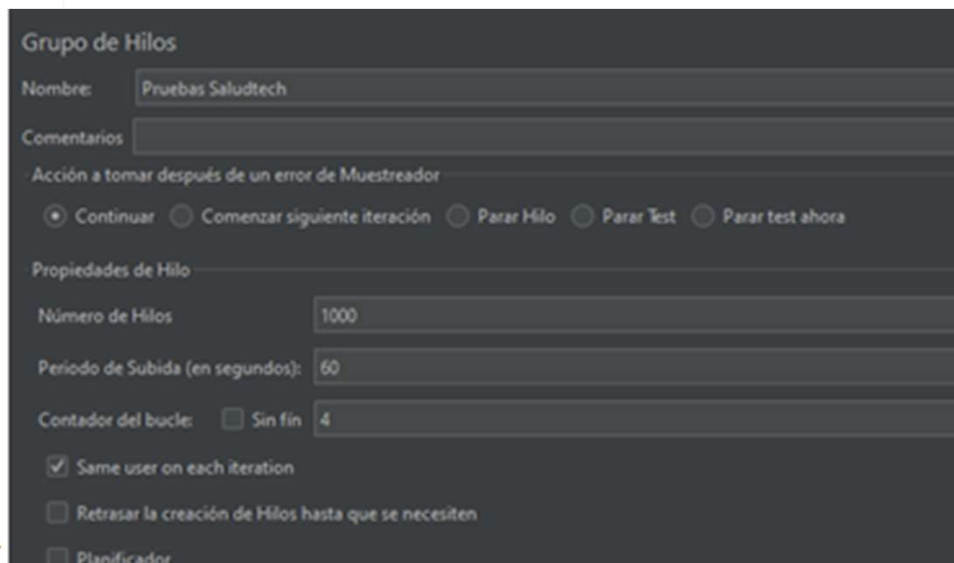
Utilizando el endpoint del BFF: '/procesar/procesar-imagen', methods=['POST']

Header:

Content-Type = json/application

Body

```
{
  "id_imagen": "46769a88-00000-4468o-8a14-1579887d01",
  "modalidad": "Rayos X",
  "patologia": "Neumonía",
  "region_anatomica": "Cráneo",
  "formato_imagen": "DICOM",
  "fuente_de_datos": "Hospital del norte",
  "antecedentes": "Diabético",
  "id_paciente": "46769a88-181f-4468-8i34-15f750381d01",
  "fecha_ingesta": "2024-02-01 09:10:00"
}
```



Grupo de Hilos

Nombre:

Comentarios

Acción a tomar después de un error de Muestreador

☒ Continuar ☐ Comenzar siguiente iteración ☐ Parar Hilo ☐ Parar Test ☐ Parar test ahora

Propiedades de Hilo

Número de Hilos

Periodo de Subida (en segundos):

Contador del bucle: ☐ Sin fin

☒ Same user on each iteration

☐ Retrasar la creación de Hilos hasta que se necesiten

☐ Planificador

# Evidencia Jmeter

Petición HTTP

Nombre:

Comentarios

Basic Advanced

Servidor Web

Protocolo:  Nombre de Servidor o IP:  Puerto:

Petición HTTP

Ruta:  Codificación del contenido:

☐ Redirigir Automáticamente ☒ Seguir Redirecciones ☒ Utilizar KeepAlive ☐ Usar 'multipart/form-data' para HTTP POST ☐ Cabeceras compatibles con navegadores

Parameters **Body Data** Files Upload

```
1 {  
2   "id_imagen": "46759a88-00000-48880-8a14-1579887d01",  
3   "modalidad": "Rayos X",  
4   "patologia": "Neumonía",  
5   "region_anatomica": "Cráneo",  
6   "formato_imagen": "DICOM",  
7   "fuente_de_datos": "Hospital del norte",  
8   "antecedentes": "Diabético",  
9   "id_paciente" : "46769a88-181f-4468-8i34-15f750381d01"  
10 }
```



## Reporte resumen

## Comentarios

· Escribir todos los datos a Archivo

Nombre de archivo

Navegar...

Log/Mostrar sólo: ☐ Escribir en Log Sólo Errores ☐ Éxitos [Configurar](#)

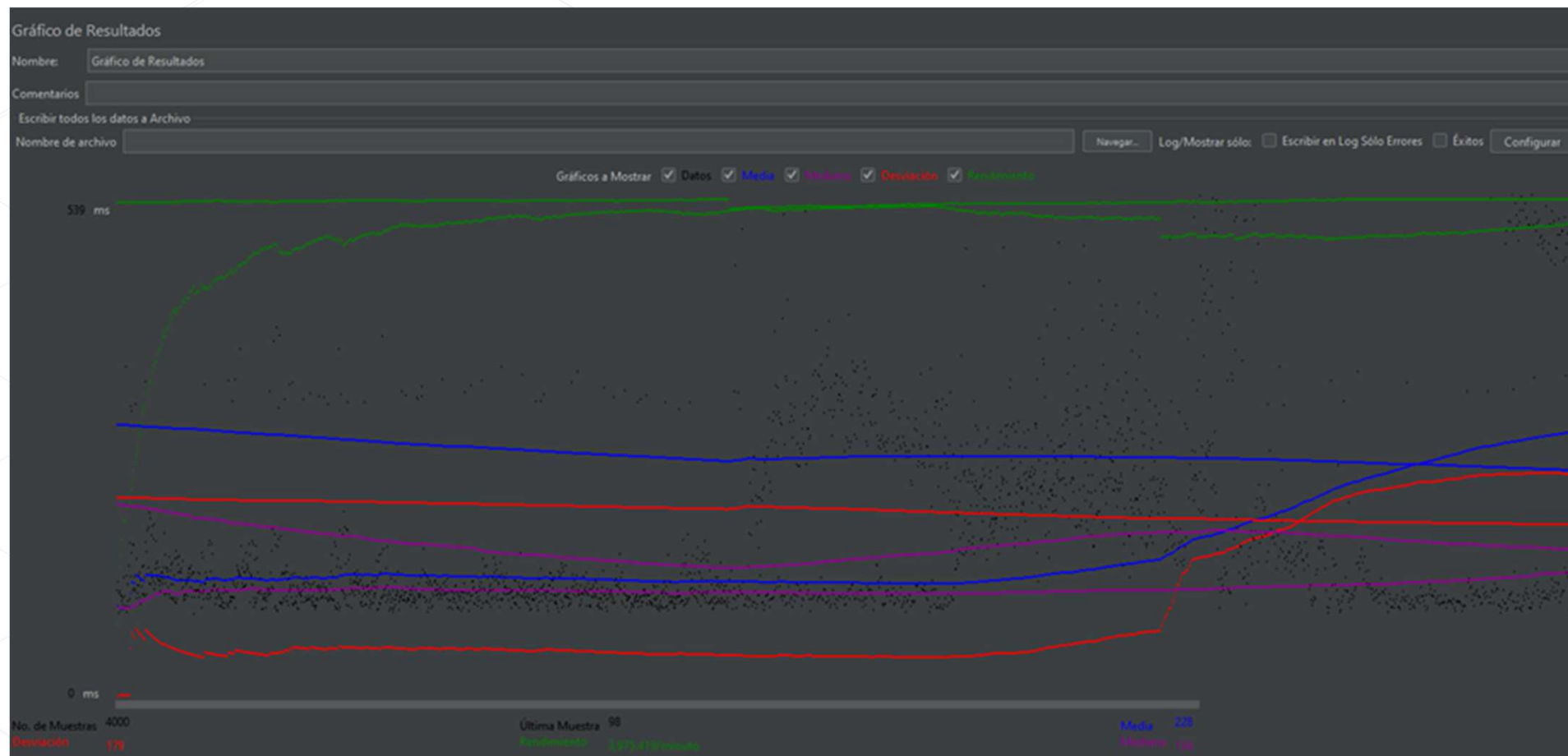
Configurar

[illegible]



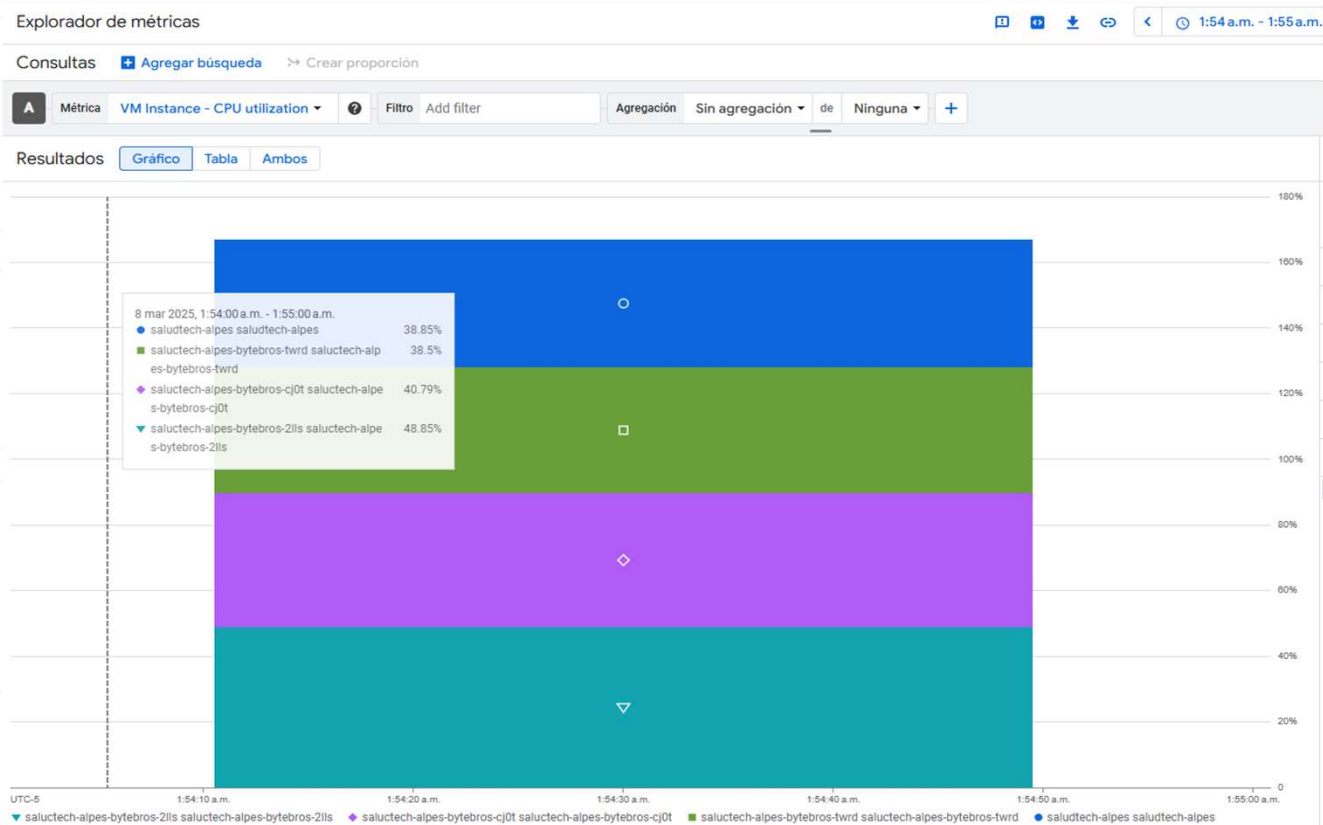
**MISO**

Maestría en Ingeniería de Software



## Evidencias

1- Presente evidencias de los resultados obtenidos en el experimento.



# Evidencia Balanceador de carga GCP

← Detalles del balanceador de cargas [EDITAR](#) [BORRAR](#)

## saludtech-alpes-lb

Balanceador de cargas de aplicaciones externo global

💡 Rendimiento web más rápido y mayor protección web con Cloud CDN y Cloud Armor. [Más información](#)

[IGNORAR](#)

[DETALLES](#)

[MONITORING](#)

[ALMACENAMIENTO EN CACHE](#)

### Frontend

Protocolo ↑	IP:Puerto	Certificado	Mapa de certificados	Política de SSL	Nivel de red ⓘ	Tiempo de espera keepalive de HTTP ⓘ
HTTP	34.54.118.91:80	-			Premium	610 segundos

### Normas de enrutamiento

Hosts ↑	Rutas	Backend
Todos los que no coincidan (predeterminado)	Todos los que no coincidan (predeterminado)	microservicios-backend

### Backend

#### Servicios de backend

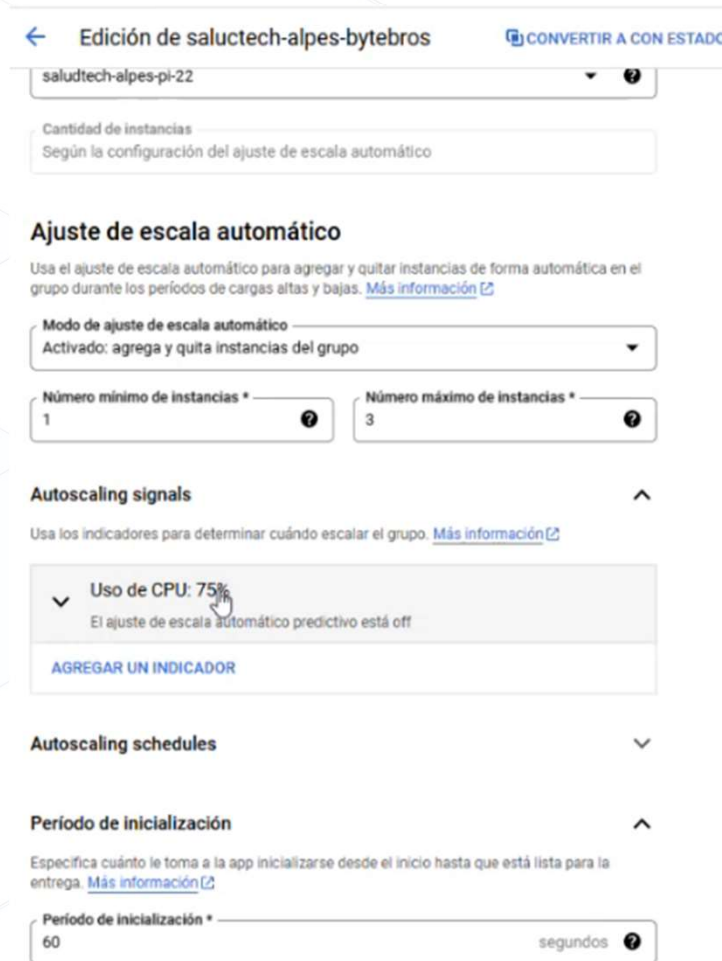
##### 1. microservicios-backend

Protocolo de extremo	HTTP
Puerto con nombre	http
Tiempo de espera	30 segundos
Política de selección de direcciones IP ⓘ	Solo IPv4
Verificación de estado	<a href="#">microservicios-saludtech</a>
Cloud CDN	Habilitada <a href="#">VER DETALLES</a>
Registros	Inhabilitada
Política de seguridad del backend	Ninguna

✓ [MOSTRAR OPCIONES AVANZADAS](#)

# Evidencia grupo de contenedores en GCP

- Se creó el grupo de contenedores en GCP, un grupo de mínimo 1, máximo 3, como regla que al alcanzar el 75% de capacidad de CPU, se redirigiera a otra instancia. Para este grupo de instancias, se usaba una plantilla de instancia con las instrucciones para correr el proyecto.



← Edición de saludtech-alpes-bytebros [CONVERTIR A CON ESTADO](#)

saludtech-alpes-pi-22

Cantidad de instancias  
Según la configuración del ajuste de escala automático

### Ajuste de escala automático

Usa el ajuste de escala automático para agregar y quitar instancias de forma automática en el grupo durante los períodos de cargas altas y bajas. [Más información](#)

Modo de ajuste de escala automático  
Activado: agrega y quita instancias del grupo

Número mínimo de instancias \* 1 [?](#) Número máximo de instancias \* 3 [?](#)

### Autoscaling signals

Usa los indicadores para determinar cuándo escalar el grupo. [Más información](#)

▼ **Uso de CPU: 75%**  
El ajuste de escala automático predictivo está off

[AGREGAR UN INDICADOR](#)

### Autoscaling schedules

▼

### Período de inicialización

Especifica cuánto le toma a la app inicializarse desde el inicio hasta que está lista para la entrega. [Más información](#)

Período de inicialización \* 60 segundos [?](#)