Pruebas de Rendimiento - JMeter con Taurus

Este documento describe el procedimiento y los resultados de las pruebas de rendimiento realizadas con Taurus, utilizando scripts de JMeter, sobre una aplicación web desplegada en Azure. La aplicación ofrece servicios de conteo de letras y palabras a través de endpoints HTTP.

Análisis del entorno de Azure

La capa **Free F1 de Azure App Service** tiene limitaciones bastante estrictas:

Recurso	Límite
vCPU	Compartida (no dedicada)
RAM	1 GB
Compute Time (Tiempo de CPU)	60 min/día

Preparación del entorno

Antes de ejecutar las pruebas, es necesario preparar el entorno con los siguientes pasos:

1. **Definir los archivos de configuración:** Se crean los archivos `.yml` que configuran las pruebas en Taurus:
    1.1. `jmeter-test/jmeter.yml`
2. **Cargar los datos de prueba:** Se utiliza un archivo CSV (`./data/data.csv`) que contiene datos variables representativos del uso real. Las variables clave son `Texto` y `Cant`.
3. **Ejecutar Taurus desde Docker:** Es necesario contar con Docker Engine en funcionamiento. La ejecución se realiza desde la carpeta del proyecto:
    3.1. `cd stress-tests`

3.2. `docker run -it --rm -v "${PWD}:/bzt-configs" blazemeter/taurus jmeter-test/jmeter.yml`

Taurus genera un enlace a BlazeMeter para visualizar métricas en tiempo real, aunque se requiere una cuenta premium para acceder a ciertos informes detallados.

Adicionalmente, es posible intensificar la carga ejecutando el mismo comando en múltiples terminales de forma simultánea.

Aplicación bajo prueba

URL del Despliegue:
`https://iyc-taller-ayajggbwd5f0d8gx.brazilsouth-01.azurewebsites.net/`

Endpoints Evaluados:

- `/static/index.html` – Página principal
- `/cantidad_letras` – POST para contar letras
- `/cantidad_palabras` – POST para contar y analizar palabras

Tipos de pruebas realizadas

1. Prueba de Carga - Funcionamiento Normal

- **Usuarios concurrentes:** 20
- **Ramp-up:** 5 segundos
- **Duración:** 20 segundos
- **Objetivo:** Evaluar rendimiento bajo condiciones normales.
- **Expectativa:** Tiempos estables, sin errores (<1%).

2. Prueba de Carga - Funcionamiento Límite

- **Usuarios concurrentes:** 50
- **Ramp-up:** 10 segundos

- **Duración:** 30 segundos
- **Objetivo:** Identificar el límite de funcionamiento óptimo.
- **Expectativa:** Leve degradación, errores bajos (<5%).

3. Prueba de Estrés Incremental

Consiste en cuatro pasos sucesivos, incrementando progresivamente la carga.

Pasos	Usuarios	Ramp-Up	Duración
1	50	10 s	30 s
2	60	10 s	30 s
3	70	10 s	30 s
4	100	15 s	30 s

Objetivo: Determinar el punto de quiebre y la capacidad de recuperación del sistema.

Expectativas por etapa:

- **Paso 1:** Degradación moderada, aún funcional
- **Paso 2:** Aumento de latencia, errores ocasionales
- **Paso 3:** Carga significativa, errores frecuentes
- **Paso 4:** Posible colapso del sistema

Métricas clave monitoreadas

1. **CPU Time:** Mide el tiempo total de CPU utilizado por el sistema para procesar las solicitudes. Esta métrica permite detectar cuellos de botella relacionados con el procesamiento interno y evaluar la eficiencia del uso de recursos.

2. **Requests:** Cantidad total de solicitudes procesadas por segundo. Este valor permite medir el rendimiento del sistema en términos de carga soportada y analizar cómo se comporta frente a distintos niveles de concurrencia.

3. **Data I/O:** Volumen de datos transferidos entre el cliente y el servidor (Input/Output). Esta métrica ayuda a entender la carga de red generada por la aplicación y su impacto en el rendimiento general.

Conclusiones

Las pruebas realizadas permiten identificar:

- El comportamiento de la aplicación bajo diferentes niveles de carga.
- La capacidad máxima de usuarios concurrentes que puede manejar.
- El punto de degradación y de quiebre del sistema.
- Recomendaciones para ajustar la infraestructura y configurar alertas de monitoreo.

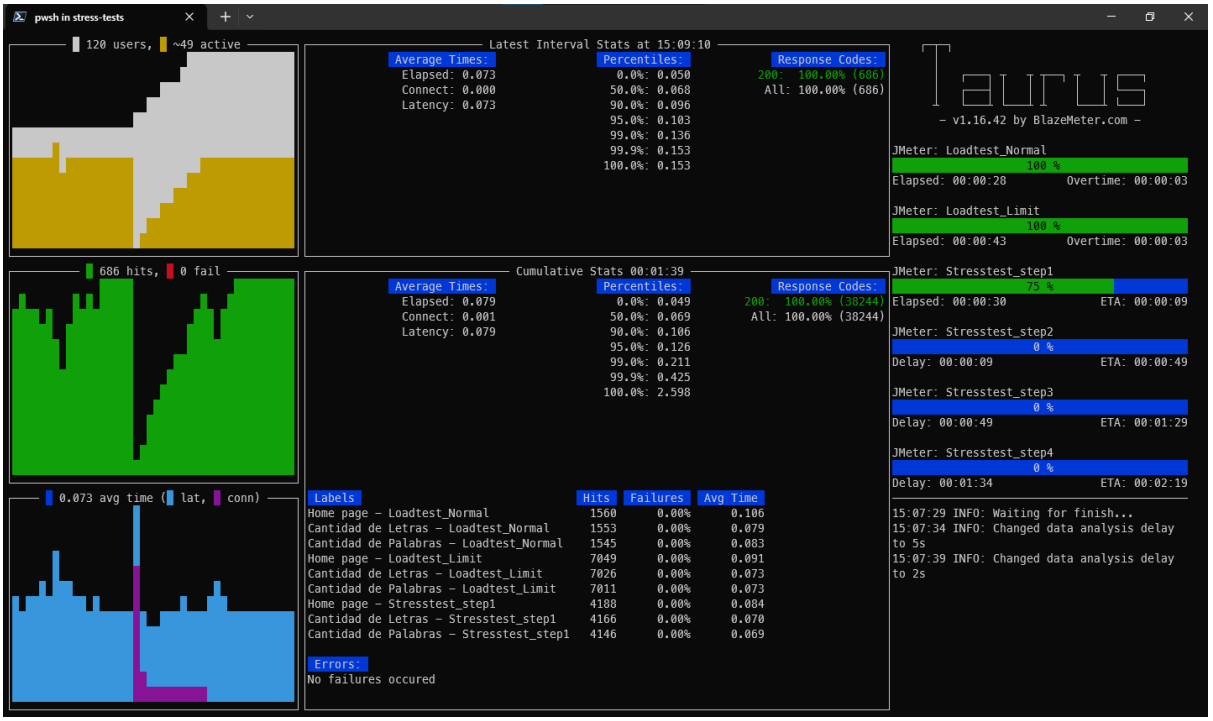
Como conclusión, haber hecho este experimento fue algo muy bueno para mi porque me permitió conocer un mundo nuevo, y aprender nuevas herramientas.

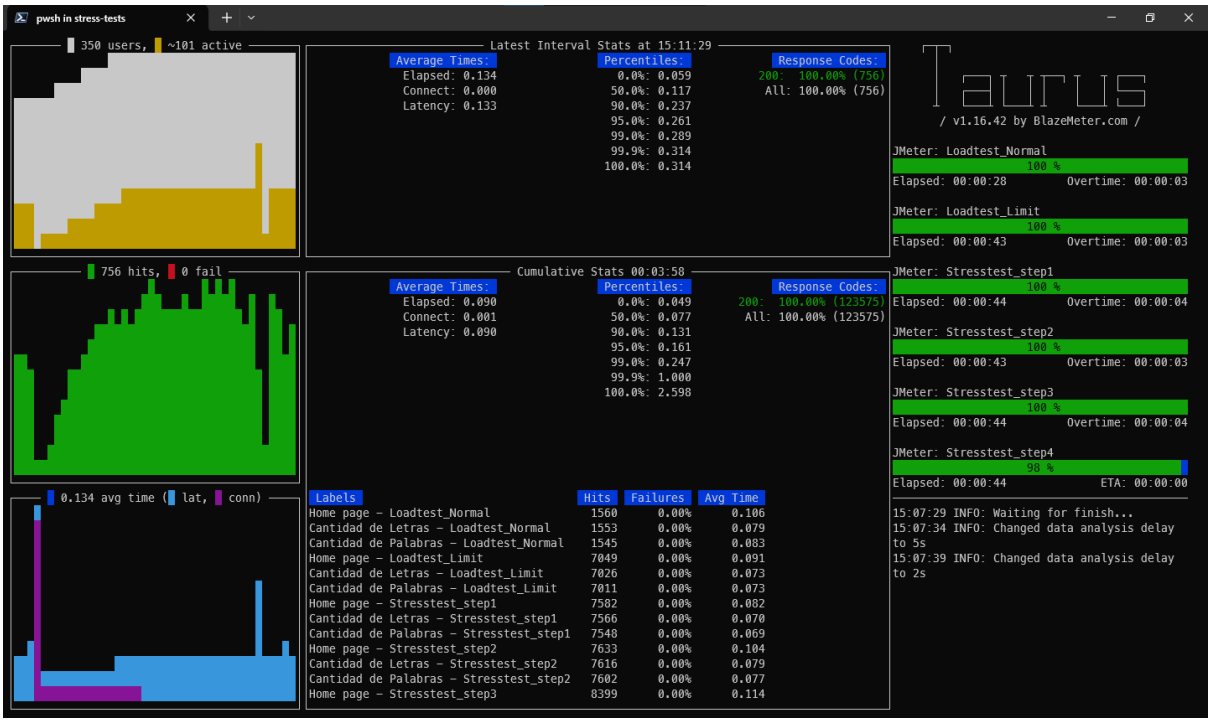
También, a pesar de las restricciones de Azure, puedo concluir que el plan gratuito es bastante resistente y completo, ya que soporta muchos usuarios concurrentes (+1000) y muchas solicitudes en cortos periodos de tiempo (+100 mil).

He de aclarar que no puse demasiada exigencia a mis pruebas. Ahora viendo los resultados obtenidos puedo decir que se podría haber estresado mucho más al sistema con un número mucho mayor de solicitudes. Tengo la hipótesis de que el punto de quiebre de mi sistema es cuando se llegan a los 60 minutos de uso de CPU diarios, con las pruebas realizadas solo llegue a cumplir poco más de 5 minutos.

Anexo

Capturas de la ejecución de las pruebas

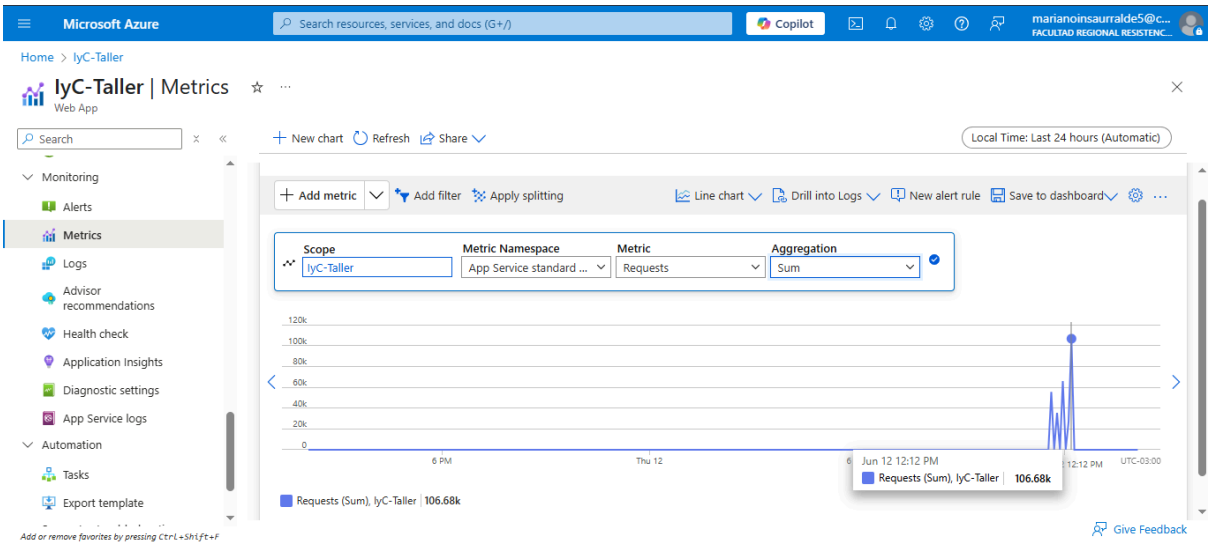




Resultados obtenidos por consola:

```
15:11:37 INFO: Post-processing...
15:11:37 INFO: Test duration: 0:04:10
15:11:37 INFO: Samples count: 133553, 0.00% failures
15:11:37 INFO: Average times: total 0.091, latency 0.091, connect 0.001
15:11:37 INFO: Percentiles:
+---------------+---------------+
| Percentile, % | Resp. Time, s |
+---------------+---------------+
|           0.0 |         0.049 |
|          50.0 |         0.078 |
|          90.0 |         0.133 |
|          95.0 |         0.163 |
|          99.0 |         0.246 |
|          99.9 |         0.706 |
|         100.0 |         2.598 |
+---------------+---------------+
15:11:37 INFO: Request label stats:
+-----------------------------------------+--------+---------+--------+-------+
| label                                   | status |    succ | avg_rt | error |
+-----------------------------------------+--------+---------+--------+-------+
| Cantidad de Letras - Loadtest_Limit     |   OK   | 100.00% |  0.073 |       |
| Cantidad de Letras - Loadtest_Normal    |   OK   | 100.00% |  0.079 |       |
| Cantidad de Letras - Stresstest_step1   |   OK   | 100.00% |  0.070 |       |
| Cantidad de Letras - Stresstest_step2   |   OK   | 100.00% |  0.079 |       |
| Cantidad de Letras - Stresstest_step3   |   OK   | 100.00% |  0.082 |       |
| Cantidad de Letras - Stresstest_step4   |   OK   | 100.00% |  0.092 |       |
| Cantidad de Palabras - Loadtest_Limit   |   OK   | 100.00% |  0.073 |       |
| Cantidad de Palabras - Loadtest_Normal  |   OK   | 100.00% |  0.083 |       |
| Cantidad de Palabras - Stresstest_step1 |   OK   | 100.00% |  0.069 |       |
| Cantidad de Palabras - Stresstest_step2 |   OK   | 100.00% |  0.077 |       |
| Cantidad de Palabras - Stresstest_step3 |   OK   | 100.00% |  0.082 |       |
| Cantidad de Palabras - Stresstest_step4 |   OK   | 100.00% |  0.093 |       |
| Home page - Loadtest_Limit              |   OK   | 100.00% |  0.091 |       |
| Home page - Loadtest_Normal             |   OK   | 100.00% |  0.105 |       |
| Home page - Stresstest_step1            |   OK   | 100.00% |  0.082 |       |
| Home page - Stresstest_step2            |   OK   | 100.00% |  0.104 |       |
| Home page - Stresstest_step3            |   OK   | 100.00% |  0.115 |       |
| Home page - Stresstest_step4            |   OK   | 100.00% |  0.151 |       |
+-----------------------------------------+--------+---------+--------+-------+
15:11:37 INFO: Sending remaining KPI data to server...
15:11:38 INFO: Ending data feeding...
15:11:40 INFO: Online report link: https://a.blazemeter.com/app/?public-token=Kt0
15:11:40 INFO: Artifacts dir: /tmp/artifacts
15:11:40 INFO: Done performing with code: 0
 mariano  stress-tests  main ≡ ~2  4m 34.265s
```

Resultados obtenidos por el Portal de Azure:



Procesando los datos de las métricas obtenidas, cree los siguientes gráficos:

Gráfico 1: Tiempo de CPU

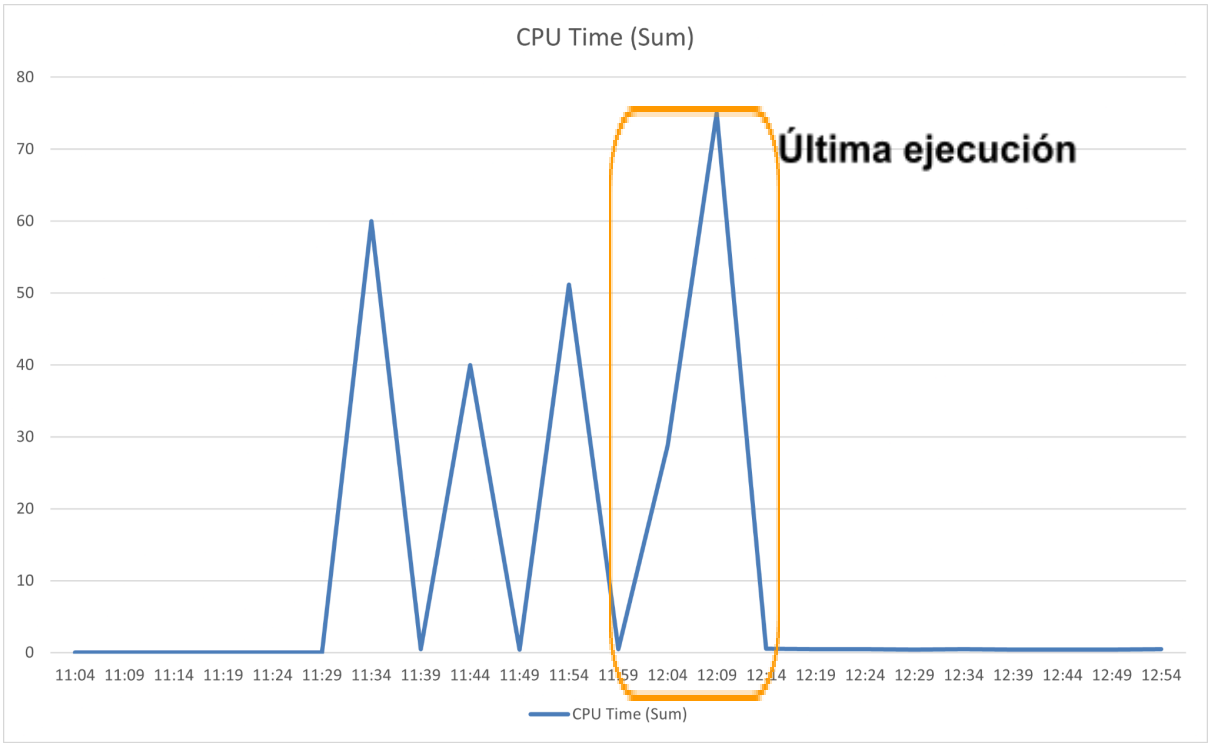


Gráfico 2: Solicitudes

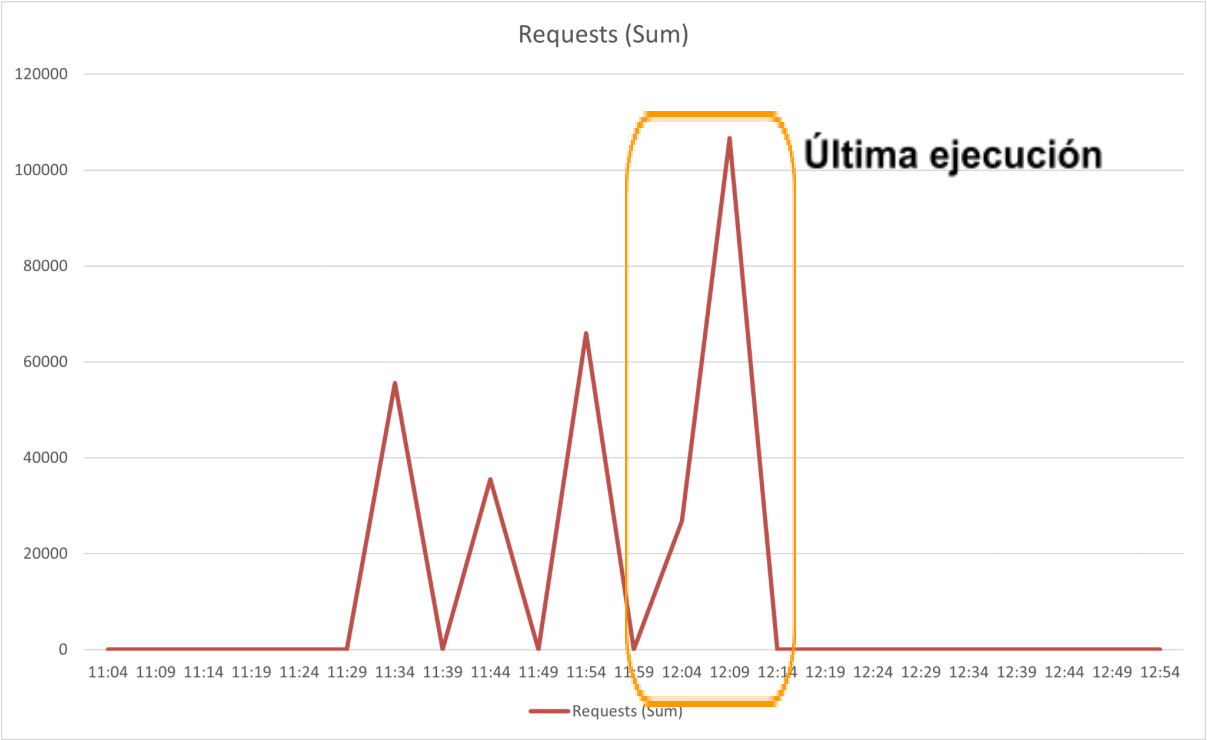


Gráfico 3: Entrada y salida de datos