Informe de Microproyecto

Tema: Balanceo de carga con HAProxy y descubrimiento dinámico con Consul
Curso: Computación en la nube
Estudiante: Daniel Muñoz
Fecha: 28 de agosto de 2025

1. Introducción

En entornos distribuidos es fundamental garantizar la alta disponibilidad y la tolerancia a fallos. Para este microproyecto se implementó una arquitectura con HAProxy como balanceador de carga y Consul como herramienta de descubrimiento de servicios. El objetivo fue desplegar múltiples instancias de una aplicación web en clienteUbuntu, registrarlas en Consul y balancear el tráfico desde servidorUbuntu mediante HAProxy. Posteriormente, se realizaron pruebas de carga con Artillery para evaluar el rendimiento en condiciones normales y bajo fallos parciales.

2. Desarrollo

La arquitectura consistió en un nodo servidor (servidorUbuntu) con Consul Server y HAProxy, y un nodo cliente (clienteUbuntu) con tres instancias de aplicación Node.js (puertos 3000, 3001 y 3002) registradas en Consul Client. Consul se configuró con verificaciones de salud en /health, mientras que HAProxy utilizó server-template enlazado al DNS de Consul para balancear tráfico dinámicamente. También se configuró una página personalizada de error 503.

3. Resultados

Escenario	Requests	Responses	RPS Prom.	p50 (ms)	p95 (ms)	p99 (ms)	Fallos VUs	HTTP 5xx
Prueba base	2100	1937	~498	498	4965	9417	163	0
/health	700	700	~9	8.9	25.8	87.4	0	0
Fallo parcial	1500	1500	~17	16.9	232.8	685.5	0	0

4. Análisis

El sistema funcionó correctamente en modo tolerancia a fallos: al detener web-3001, Consul detectó el cambio y HAProxy redistribuyó el tráfico a las otras instancias. En la prueba base, la alta carga generó latencias elevadas y 163 fallos, mostrando límites de capacidad del sistema. El endpoint /health respondió rápido y sin fallos. Durante la simulación de falla parcial no hubo errores HTTP ni usuarios fallidos, solo un incremento temporal de latencia.

5. Conclusiones

1. La integración Consul + HAProxy permitió implementar un balanceador dinámico y tolerante a fallos. 2. El sistema responde adecuadamente en escenarios de fallo parcial, manteniendo la disponibilidad. 3. El endpoint /health asegura la detección rápida de servicios disponibles. 4. Bajo cargas extremas, la latencia se incrementa significativamente. 5. La personalización de la página 503 mejora la robustez frente a caídas totales.