Balanceador de Carga con NGINX

Autor: Krsna Gutiérrez González (@Krsz1)

Asignatura: Servicios Telemáticos

Fecha: 11 de noviembre de 2025

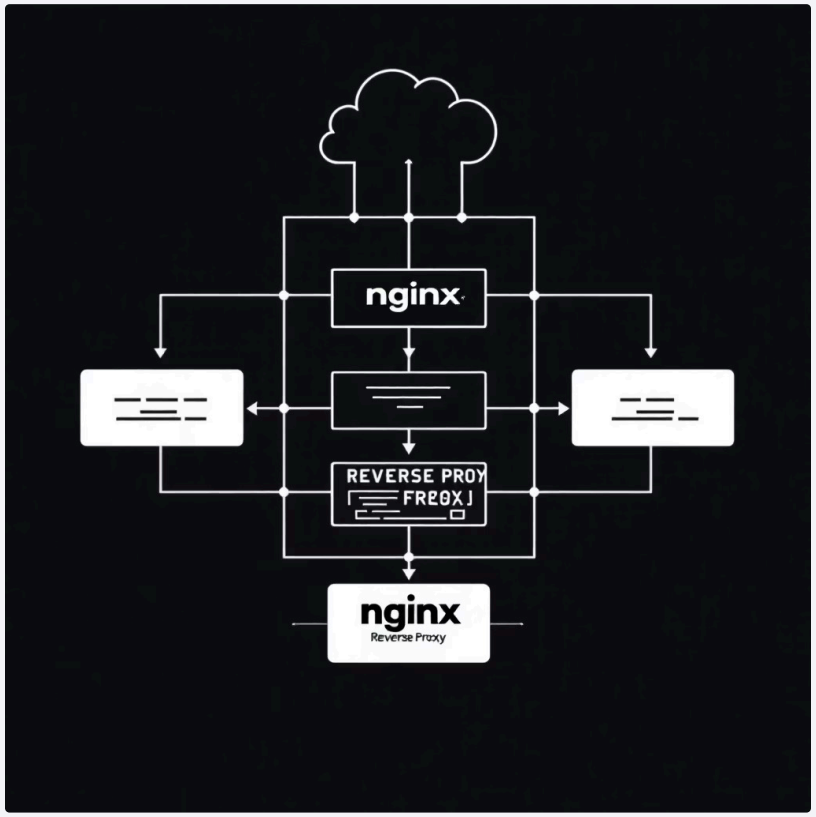
Repositorio: github.com/Krsz1/Servicios-telematicos

Descripción del proyecto

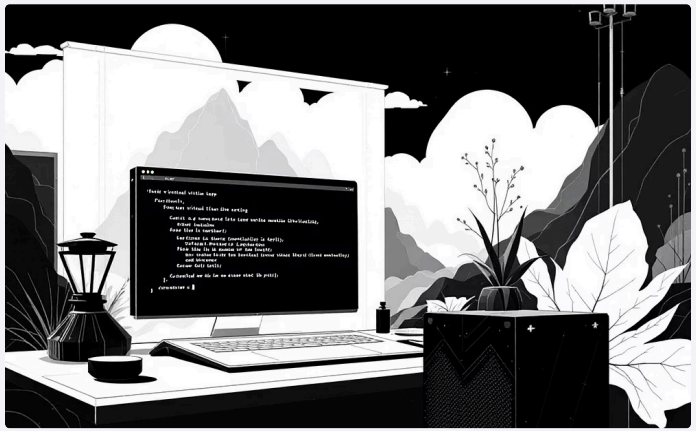
Proyecto educativo que implementa un sistema de balanceo de carga con NGINX para mejorar disponibilidad, distribución del tráfico y rendimiento de servidores web.

- 1 balanceador (NGINX)
- Hasta 4 servidores web backend (escalables)
- 1 cliente para pruebas de carga (Artillery)
- Sistema automatizado de pruebas y análisis

Objetivo: eliminar punto único de fallo y equilibrar la carga entre servidores.

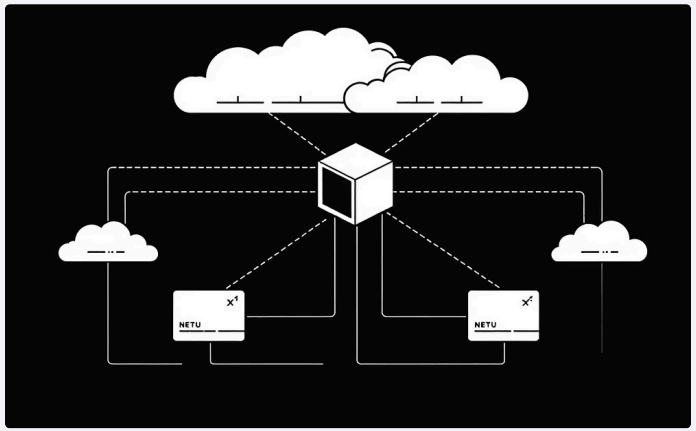


Automatización: Vagrant + VirtualBox



Por qué Vagrant

Automatiza la creación y configuración del entorno, reproducible y portable entre sistemas operativos.



Topología levantada

5 máquinas virtuales: lb, web1, web2, web3, cliente de pruebas. Red privada y port forwarding configurados.

Automatizaciones realizadas

- Instalación automática de NGINX en lb y web*
- Instalación Node.js + Artillery en cliente
- Configuración de redes y servicios

Conclusiones técnicas

NGINX

Distribuye la carga de forma eficiente y robusta, con salud de backend y failover automático.

Artillery

Simula patrones de tráfico realistas (baseline, ramp, spike) y ofrece datos accionables.

Vagrant

Permite reproducir el laboratorio completo y escalar servidores de forma controlada.

Procesamiento

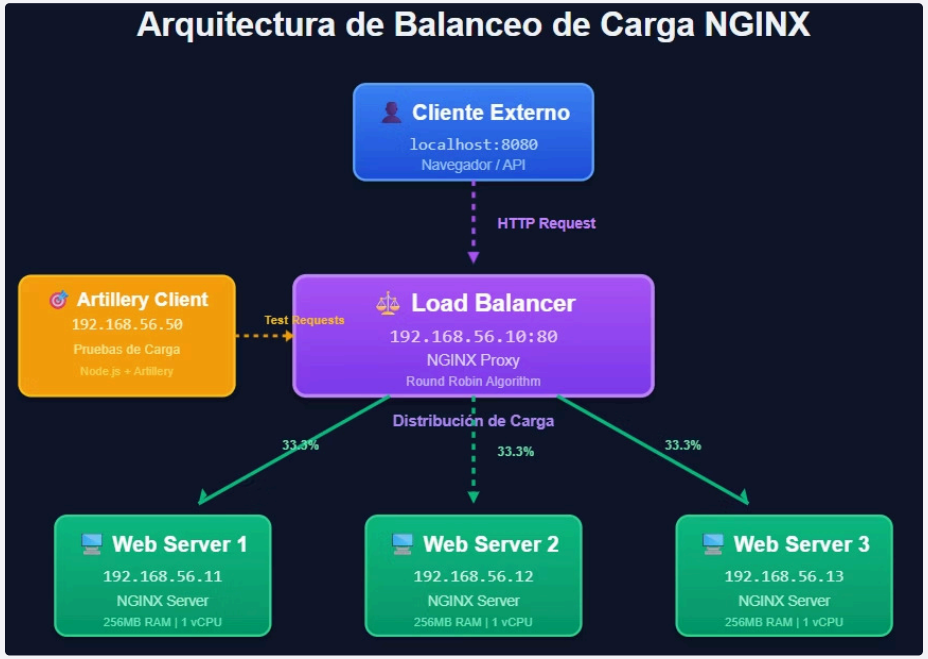
Bash + jq automatizan la extracción de métricas y generación de reportes comparativos.

Resultado: un flujo completo desde la simulación hasta la interpretación que facilita decisiones de arquitectura y tuning.

Diseño de la arquitectura y direcciones IP

Despliegue en red privada VirtualBox. Acceso host → lb mediante localhost:8080 (port forwarding).

Elemento	Dirección / Notas
Load Balancer (NGINX)	192.168.56.10 — reverse proxy, Round-Robin, health checks, puerto 80→8080
web1 / web2 / web3 / web4	192.168.56.11–14 — NGINX, 256 MB RAM (web4 opcional)
Cliente de pruebas	192.168.56.50 — Node.js 20.x, Artillery 2.0.20, scripts bash, 512 MB RAM



Pruebas de Carga con Artillery

Para ejecutar las pruebas de rendimiento y estrés, conéctese al cliente y acceda a los escenarios predefinidos:

```
vagrant ssh client
cd /vagrant/artillery
```

Carga Constante (Baseline)

Simula una carga nominal para establecer una línea base de rendimiento y estabilidad.

- Duración: 60 segundos
- Tasa: 50 peticiones/segundo
- Esperado: Sistema estable, latencia <50ms, 0% errores

Carga Incremental (Ramp)

Aumenta progresivamente el volumen de tráfico para identificar límites de capacidad.

- Fase 1: 120s @ 10 req/s (calentamiento)
- Fase 2: 240s @ 50 req/s (normal)
- Fase 3: 240s @ 100 req/s (crecimiento)
- Fase 4: 120s @ 200 req/s (alta carga)

Pico de Tráfico (Spike)

Evalúa la resiliencia del sistema ante un aumento súbito y extremo de usuarios.

- Fase 1: 30s @ 500 req/s (pico extremo)
- Fase 2: 90s @ 50 req/s (recuperación)
- Esperado: Errores durante el pico, recuperación posterior

Cada escenario se ejecuta con diferentes configuraciones de servidores web para analizar el escalado:

- Pruebas con 2 Servidores
- Pruebas con 3 Servidores (Configuración por Defecto)
- Pruebas con 4 Servidores
- Pruebas con 1 Servidor (Escenario Extremo)

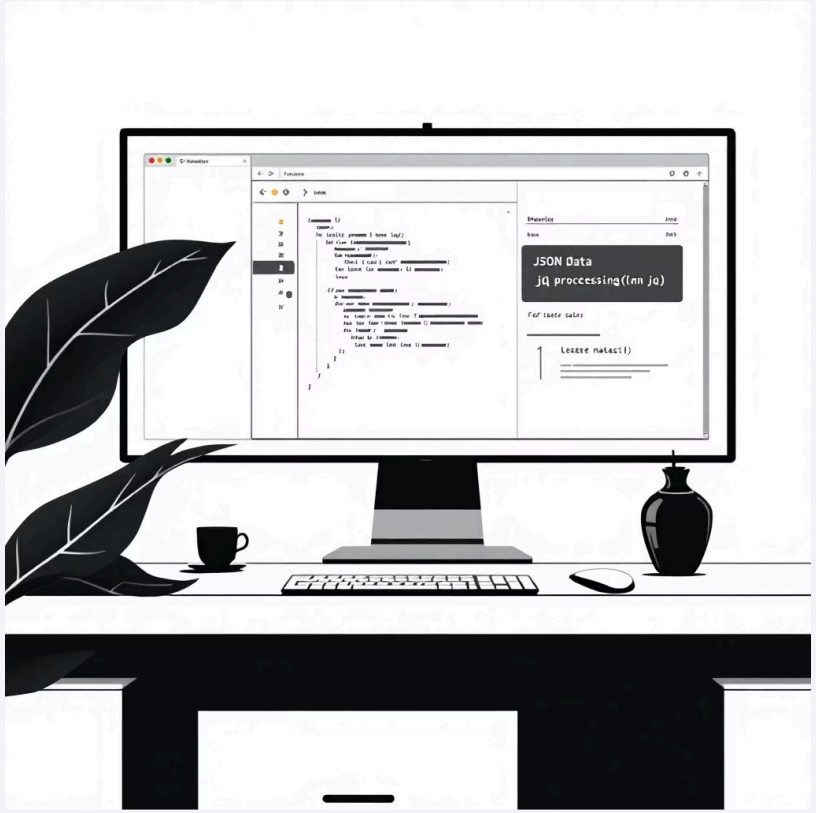
Sistema de generación de reportes

Scripts principales:

- **generate_report.sh** — procesa resultados y genera reportes Markdown con métricas: peticiones, usuarios, tasa de éxito, mean, median, p95, p99, min, max.
- **generate_comparative_report.sh** — compara escenarios y configuraciones (nº de servidores, tipo de prueba).

Flujo de análisis:

1. Ejecutar Artillery → JSON
2. Procesar JSON con jq y generate_report.sh
3. Generar reportes individuales y comparativos; extraer conclusiones



Referencias

Documentación Oficial y Técnica

- Artillery.io. (2024). Artillery documentation. **https://www.artillery.io/docs**
- DigitalOcean. (2024). How to set up Nginx load balancing. DigitalOcean Community Tutorials. **https://www.digitalocean.com/community/tutorials/how-to-set-up-nginx-load-balancing**
- Gatling Corp. (2024). Simulation setup. Gatling Documentation. **https://gatling.io/docs/gatling/reference/current/core/simulation/**
- HashiCorp. (2024). Vagrant documentation. **https://www.vagrantup.com/docs**
- NGINX. (2024a). NGINX documentation. **https://nginx.org/en/docs/**
- NGINX. (2024b). HTTP load balancing. NGINX Documentation. **https://nginx.org/en/docs/http/load_balancing.html**

Artículos Científicos y Académicos

Load Balancing y Algoritmos

- Alakeel, A. M. (2010). A guide to dynamic load balancing in distributed computer systems. International Journal of Computer Science and Network Security, 10(6), 153-160.
- Kansal, N. J., & Chana, I. (2012). A comparative study of load balancing algorithms in cloud computing. International Journal of Computer Applications, 47(20), 14-19. **https://doi.org/10.5120/8488-2370**
- Kumar, P., & Kumar, R. (2016). Performance analysis of load balancing algorithms in distributed system. International Journal of Computer Applications, 135(4), 35-39. **https://doi.org/10.5120/ijca2016908215**
- Mondal, B., Dasgupta, K., & Dutta, P. (2012). Load balancing techniques: Major challenges in cloud computing. In 2012 International Conference on Computing Sciences (pp. 132-137). IEEE. **https://doi.org/10.1109/ICCS.2012.14**
- Nishant, K., Sharma, P., Krishna, V., Gupta, C., Singh, K. P., & Rastogi, R. (2011). Load balancing of nodes in cloud using ant colony optimization. In 2011 14th International Conference on Computer Modelling and Simulation (pp. 3-8). IEEE. **https://doi.org/10.1109/UKSim.2011.56**
- Sharma, S., Singh, S., & Sharma, M. (2008). Performance analysis of load balancing algorithms. World Academy of Science, Engineering and Technology, 38(3), 269-272.
- Solanki, P., & Patel, S. (2015). Performance evaluation of web servers using central load balancing policy over round robin. International Journal of Computer Applications, 125(9), 15-20. **https://doi.org/10.5120/ijca2015906089**