Witchly scaling case study

Case Study: Scaling Witchly.host to 10k Users on Bare Metal

Project: Witchly.host (Game Hosting ISP)
Role: Founder & Lead Systems Engineer
Status: Bootstrapped | Profitable | 10,000+ Registered Users

🚀 Executive Summary

The Goal: Architect a high-performance game hosting infrastructure capable of supporting 10,000+ users with sub-20ms latency.
The Constraint: Zero VC funding. Required extreme capital efficiency (<€150/mo OpEx).
The Result: Built a distributed bare-metal system that handled 50Gbps+ DDoS attacks and thousands of concurrent containers.

---

1. The "Lean" Tech Stack

Instead of using expensive managed cloud solutions (AWS/GCP), I designed a **heterogeneous bare-metal cluster** to maximize raw compute power per dollar.

The Hardware (Data Plane)

- **Provider:** Hetzner Bare Metal.
- **Specs per Node:** Ryzen 5 3600 (6 Cores/12 Threads), 64GB DDR4 RAM, 1TB NVMe SSD.
- **OS:** Ubuntu 20.04 LTS (Kernel tuned for high-throughput I/O).

The Orchestration

- **Virtualization:** Docker + Pterodactyl Panel.
- **Isolation:** Implemented strict resource throttling (CPU pinning & RAM limits) to prevent "Noisy Neighbor" issues common in shared hosting.

- **Customization:** Deployed custom 'Eggactyl' configurations to support niche game environments requested by clients.

The Automation

- **Custom Bot:** Developed a Discord Bot (Python/Node.js) to handle the "Order-to-Provision" pipeline.
- **Billing:** Integrated Razorpay & PayPal APIs via Slash Commands for instant server activation.

---

2. Architecture Diagram (Conceptual)

- **Frontend (User Facing):** WordPress (VPS) → **Cloudflare** (CDN/WAF).
- **Control Plane:** Pterodactyl Panel (Separate VPS) → **Cloudflare Proxy**.
- **Data Plane (Nodes):** Direct Bare Metal → **Raw UDP/TCP** (Optimized for Latency).

---

3. Engineering Spotlight: The DDoS Mitigation

The Incident:
During peak traffic, the infrastructure suffered a coordinated **L7 DDoS attack** (Mass IP Requests & TCP Floods). The Control Panel API was overwhelmed, causing crashes every 5 minutes.

The Solution:

- **Traffic Analysis:** Identified the attack vector as a volumetric botnet targeting the login API.
- **WAF Hardening:** Deployed aggressive **Cloudflare Page Rules** to cache static assets and force "Under Attack Mode" (JS Challenge) specifically for panel authentication.
- **Geo-Fencing:** Analyzed logs to pinpoint the botnet origin; blocked specific malicious ASNs and high-risk countries at the edge.
- **Local Defense:** Configured **UFW & IP Tables** on bare metal nodes to drop non-whitelisted traffic, protecting unproxied game ports.

Outcome: System stability restored in <20 minutes with zero data loss.

---

4. Key Metrics (Proof of Work)

- **Scale:** 10,000+ Registered Users.
- **Efficiency:** Achieved **~€100/month** total infrastructure cost. (Equivalent compute on AWS EC2 estimated at $3,000+/mo).
- **Performance:** Maintained low-latency connectivity for primary markets (Germany/Finland) via strategic routing.

Contact

Tushar *Systems Engineer & Backend Developer*
📍 Bangalore, India

- **Email:** tushar@tushar-a.dev
- **LinkedIn:** https://www.linkedin.com/in/tushar-a-607371315/
- **Github:** https://github.com/Witchly