Building a Stealth VPN with Claude Code

An architectural case study on adversarial network design, autonomous DevOps, and bypassing the GFW.

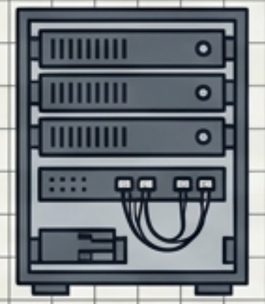
`[Self-Hosting] [DevOps] [VLESS-Reality] [AI Agents]`

> Execute Task

Objective: Build a complete, censorship-resistant proxy service with a shareable client configuration. Absolute zero commercial VPN tracking.

The Agent: Claude Code

Access: Granted raw SSH credentials to a fresh server. Full autonomous deployment.



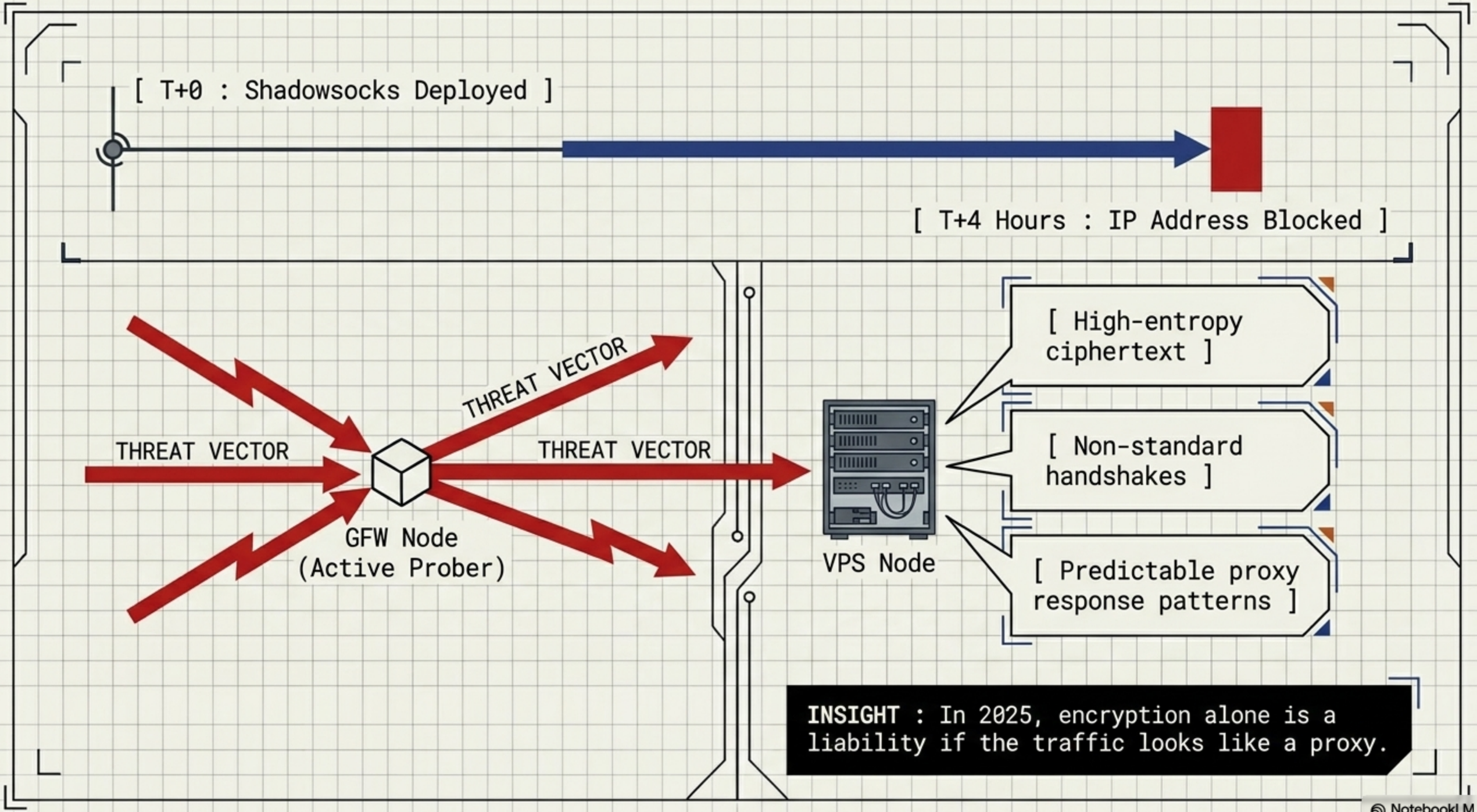
SYSTEM ENVIRONMENT

[+PROVIDER] : Atlas Networks (Los Angeles)

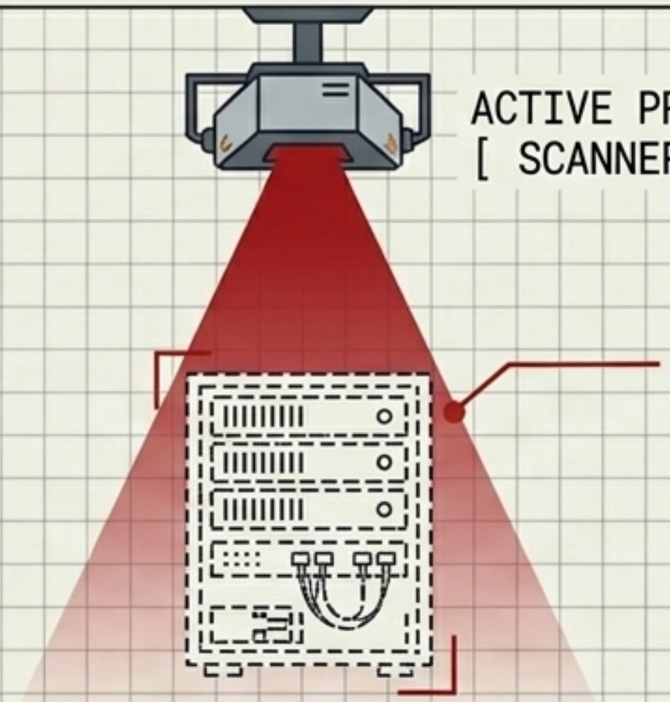
[+ROUTING] : 9929 Premium

[+NETWORK] : Static Residential IP

[+BUDGET] : ~$20 / month


[T+0 : Shadowsocks Deployed]
[T+4 Hours : IP Address Blocked]
THREAT VECTOR
THREAT VECTOR
THREAT VECTOR
THREAT VECTOR
GFW Node
(Active Prober)
VPS Node
[High-entropy ciphertext]
[Non-standard handshakes]
[Predictable proxy response patterns]
INSIGHT : In 2025, encryption alone is a liability if the traffic looks like a proxy.

Hiding as Nothing (Obfuscation)



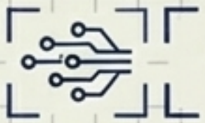
Traditional proxies strip identifying metadata. The absence of standard protocols ironically becomes a highly detectable fingerprint.

Hiding in Plain Sight (Adversarial Mimicry)

VLESS-Reality borrows the TLS fingerprint of a major, unblockable domain. The GFW faces a dilemma: allow the proxy, or block all traffic to Microsoft. It defaults to allowing.

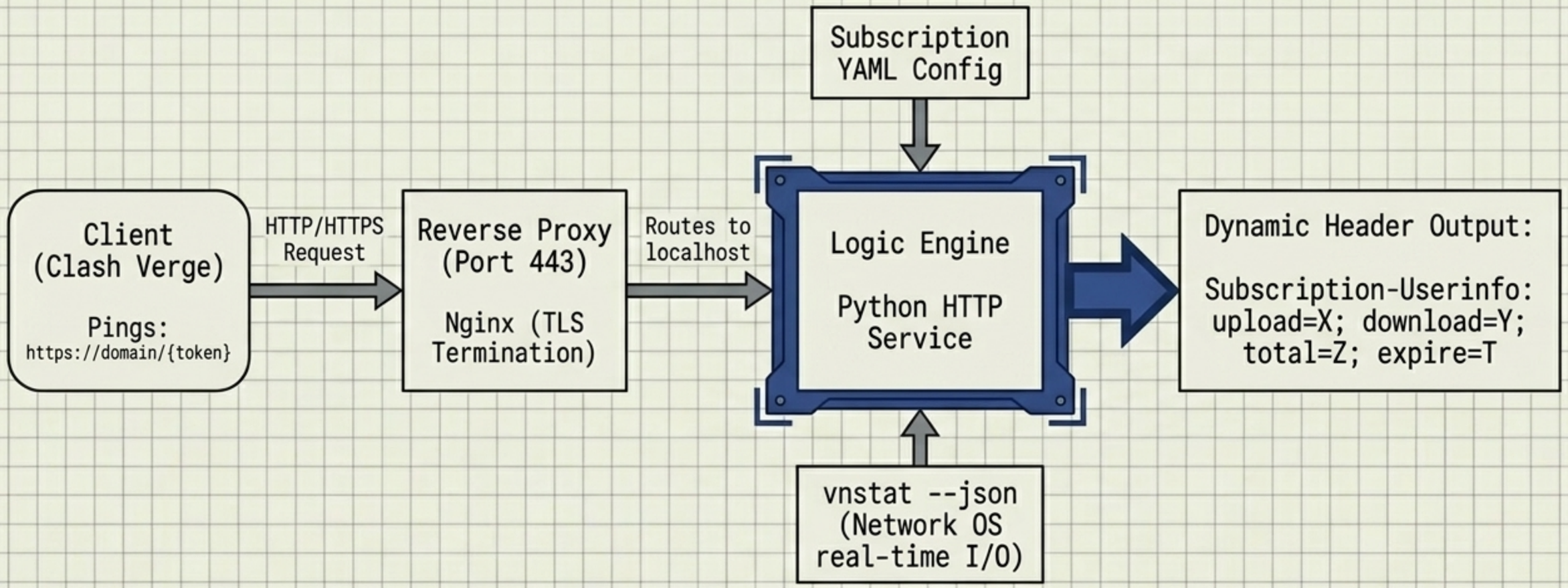
Protocol Comparison & Deployment Strategy

Protocol	Transport Layer	Evasion Strategy	GFW Resilience	Role in Stack
Shadowsocks	TCP/UDP	Obfuscation	Weak (Fingerprinted)	Discarded
VLESS-Reality	TCP (TLS)	Mimicry (Real Site TLS)	Very Strong	Primary (Auto-Select)
Hysteria2 / TUIC-v5	QUIC (UDP)	Speed / Brute-force	Medium (UDP-dependent)	Fallback
Vmess-WS	WebSocket	HTTP Proxying	Weak	Last Resort Only



Note: Auto-select routing ensures Reality handles all primary traffic invisibly.

Subscription Server Architecture & Traffic Quota Flow.



Result: Clients automatically render a live traffic quota bar (e.g., 3TB monthly limit) based on the dynamic header, requiring zero manual configuration from the end-user.

The AI Debugging Loop

The Cert Saga

Trigger: Let's Encrypt rejects certificate.

Action: Inspect acme.sh internal state.

Root Cause: Setup script cached domain in email field.

Resolution: Delete CA cache directory completely. Re-run with explicit --accountemail flag.

The Broken Traffic Bar

Trigger: Clash Verge fails to fetch traffic stats.

Action: Trace full request chain (Client -> Nginx -> Python).

Root Cause: Client uses HEAD requests; Python service lacks method.

Resolution: Implement do_HEAD handler in Python server.

Key Insight: 10-15 minute autonomous resolution cycles characterized by deep root-cause state analysis, entirely replacing trial-and-error workarounds.

Zone of Autonomy: AI vs. Human Engineering Division

ZONE OF AI MASTERY (SERVER-SIDE)

- Tasks with deterministic success criteria (e.g., URL returns valid YAML).
- Perfect execution of infrastructure configuration and stack orchestration.
- System-level state debugging.

TECHNICAL BOUNDARY: CLIENT-SIDE IDIOSYNCRASIES

THE HUMAN BOUNDARY (CLIENT-SIDE UX)

- Undocumented UI/UX edge cases.
- Client-specific behaviors (e.g., Clash Verge silently deleting manually placed config files on restart).
- Requires human observation and reasoning.

SYNTHESIS & CONCLUSION

Synthesis: AI excels at complex, deterministic engineering. Humans remain essential for observing and reasoning through idiosyncratic client behaviors. A perfect division of labor.