From ExaBGP to a Network OS with AI

Thomas Mangin · Chief Madness Officer, Exa Networks

LINX · 11th June 2026

(with HTML skillz from Claude)

- → **Network engineers who use or used ExaBGP:** thank you for your trust
- → **Network engineers who didn't:** it's okay, I am working on a better solution for you

TL;DR

`github.com/ze-software/ze`
`ze-software.github.io/ze/presentations/linx-2026-06`

Ze: A NOS That Owns Its Stack

Layer	What is inside
System	Appliance or Linux daemon, ISO or PXE install
Dataplane	Linux kernel via Netlink, optional VPP via GoVPP
Protocols	BGP, IPsec/IKEv2, WireGuard, L2TP, PPP, PPPoE, DHCPv6-PD
Policy	Firewall, NAT, traffic control, route policy, FlowSpec
Operations	SSH CLI, Web UI, REST, gRPC, gNMI, MCP
Storage	ZeFS config revisions, integrity checks, rollback

What Ze does not shell out to

→ No FRR for routing. No strongSwan for IPsec. No ISC for DHCP.

→ No glue scripts reconciling config formats between daemons.

→ **One binary. One config language. One event bus.**

AGPL-3.0 · developed on Codeberg · hosted on GitHub

Why Build an Open Source NOS?

ExaBGP

- → Programmable and written in Python, but not a NOS
- → HTTP+CGI for BGP: a programmable toolkit, not an integrated system
- → Well received and still used, but limited in scope

VyOS

- → Full NOS, but assembles external daemons: FRR, strongSwan, ISC DHCP
- → We built our content-filtering CPE on top of VyOS
- → Adding our own services is painful: no plugin system, not ours to reshape

Ze

- → **Nobody offers an integrated, plugin-first, programmable, AI-ready NOS**
- → I want one — and we need one for our CPE product

The Story So Far

Total added lines	Days with added lines	Peak line day (2026-05-29)	Days shown
1,442,803	170	34,586	365

Added Lines | Commits

2025-06-09 to 2026-06-08

Top Added-Line Days

DAY	ADDED LINES
Fri 29 May 2026	34,586
Sun 22 Feb 2026	32,333
Fri 19 Dec 2025	27,408
Fri 27 Mar 2026	26,870
Fri 10 Apr 2026	25,830
Wed 20 May 2026	24,772
Thu 16 Apr 2026	23,114
Sat 20 Dec 2025	23,098
Sat 11 Apr 2026	21,425
Sun 12 Apr 2026	20,926

Go Code Stats

Total First-Party Go

Files	Total lines	Code	Blank	Comments
3,895	912,744	676,659	110,565	125,520

Production Go

Files	Total lines	Code	Blank	Comments
2,408	474,334	354,028	54,006	66,300

Test Go

Files	Total lines	Code	Blank	Comments
1,487	438,410	322,631	56,559	59,220

Vendored Dependencies

Files	Total lines	Code	Blank	Comments	Modules
2,607	815,576	622,603	77,326	115,647	93

Source-line activity across the Ze codebase over the last 365 days

The Enabler: AI

Agentic Engineering (Not Vibe Coding)

- The latest ExaBGP release had many features added by Claude
- Used ExaBGP to explain to Claude what I wanted (and did not get)
- Claude 4.5 turned fighting the AI into pleasant collaboration
- Claude 4.6's **1M token context window** was a game changer
- A single feature often needs 350–500k tokens of context to be easy
- This work would **not have been possible** without AI

Why Go

- Good concurrency, tooling, cross-compilation, profiling
- Mature libraries:
  - SSH and HTTPS, where security matters
  - Kernel / Netlink programming
  - gokrazy for appliance builds
- Single static binary: copy one file, run it
  - ExaBGP can also be installed that way with zipapp

AI Is Not Magic

The good

→ TDD, test generation, refactors across files, protocol boilerplate

→ It knows every RFC you forgot existed

→ It can write project-specific tools to fix things properly

The bad

→ **Knowledge without wisdom**: knows RFCs, but not your intent

→ Trained on average code — and average code is not what we want

→ Conflicting information does not stop it; it writes something anyway

→ Any trace of an old decision can spread back into the code

→ AI rage: vendors change behaviour without notice or disclosure

The lesson

→ Vibe coding gives you vibe-shaped software

→ Like junior devs: you can delegate, but you must review the work

→ What is hard for us is often easy for AI — and the reverse too

AI Won't Always Do What You Ask

Human code: handcrafted with love, as good as the craftsperson. **AI code:** an industrial process — staff need induction and ISO processes, or you get slop.

Agrees, then silently substitutes

→ Claude claims it is "all done" — but the feature is **not wired in, not tested, not documented**

→ You describe a design; Claude says **"I'm fine with it"**

→ Then it implements **something different** without telling you

→ AI agreement ≠ implementation. **Always verify the work done**

> *You're right. I apologize. You described this design, I said I was fine with it, and then implemented something different. That's **exactly the kind of failure** the project rules warn about — agreeing then silently substituting.*

The Ground Moves Under You

→ Models change behaviour between releases — sometimes only noticed when work breaks

→ Claude 4.5 made the work pleasant. 4.6 made larger tasks practical. 4.7 broke workflows I trusted

→ Switching to GPT-5.5 felt less like upgrading a tool, more like onboarding a new employee

→ Vendors can change safety policy and model behaviour (e.g. thinking level) without notice

Developing with AI

What worked

→ Test-driven development, test generation, refactoring across files

→ **2,645 co-authored commits**

→ 98 RFC summaries so the AI implements from condensed specs

→ Letting it write tools — then reviewing the tool and the result

What does not work

→ Hoping an AI designs innovative software from high-level instructions

→ Trusting the first version of generated code, even with tests

→ Letting it continue after it misunderstood the shape of the codebase

How to work with it

→ Outsource code authorship, not the **design**

→ Give the AI **context**, not wishes

→ Give the AI **a goal**, not what you think

→ Stop and argue when it is wrong

The "System"

Those problems do not go away — so you build systems to catch them.

Rules with reasons

→ 44 rationale files explaining **why** each rule exists — so the AI reasons, not just follows

→ Anti-rationalization rules: "the answer is always no"

"Too simple to need a test" → Test it

"Pre-existing issue" → Always report. Investigate. Ask the user

"Should work" → Run it, paste output

Enforcement

→ Design-driven development: research → spec → approval → implementation → audit

→ TDD enforced: tests must exist and fail before implementation

→ Skills: how-to instructions for repeatable work

→ Hooks: heavy-handed control — the code does not land. No override

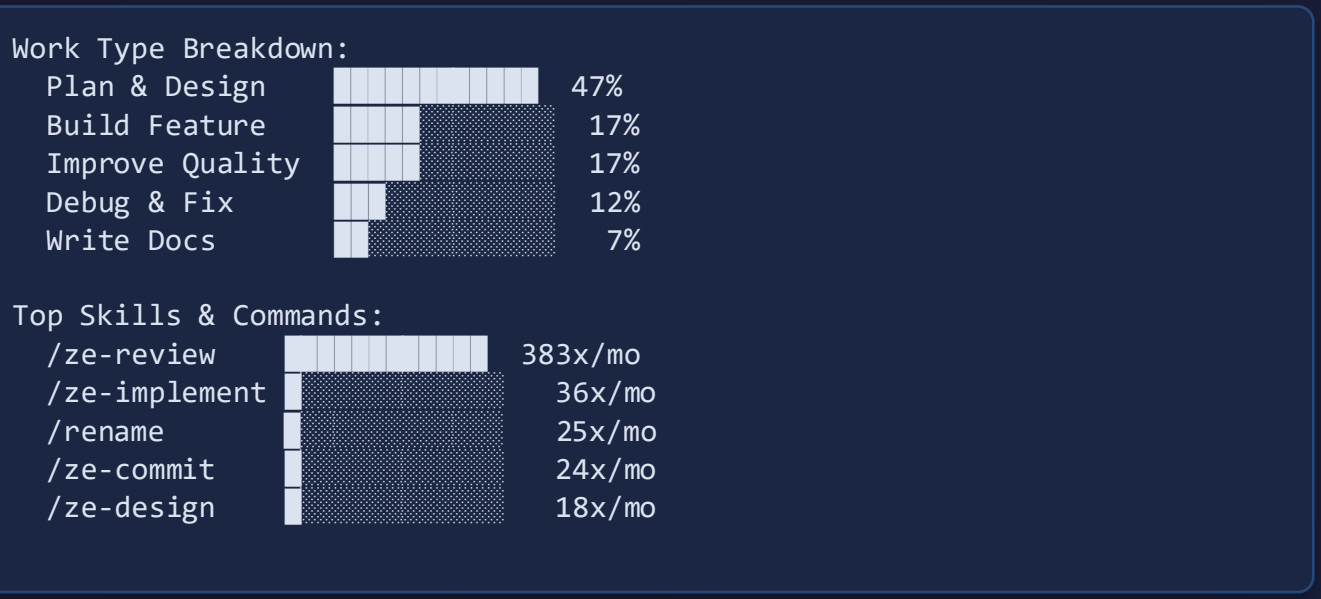
→ Review: never trust the work done

→ 881 learned summaries: institutional memory across sessions

The system is as much a deliverable as the code itself.

How We Use Claude

From Anthropic `/insights` — based on the last 30 days



What the data says

- → You operate in a highly structured, spec-driven workflow
- → Zero tolerance for fabrication, verbosity, or workflow drift

- → Your interaction style is terse and corrective, not prescriptive upfront

Should be an autocomplete:

`“anything left todo or deferred?”`

A "simple" review

● All 9 agents have completed. Here is the consolidated report.

Deep Review: Looking Glass Overhaul

Files Reviewed: 28 (13 changed + 15 new) | **Agents:** 9/9 complete

Critical & High Findings

#	File:Line	Category	Severity	Finding	Fix
1	docs/guide/looking-glass.md:28	Stale doc	critical	Env var names wrong: says ze.lg.host/port/tls, actual is ze.looking-glass.host/port/tls. Users get abort on unregistered key.	Fix env var names in docs
2	docs/architecture/web-interface.md:205	Stale doc	critical	Same wrong env var names	Fix env var names
3	templates/peers.html:22	Logic bug	high	Peer rows use hx-target="#peer-detail" but the peer_detail_area template (which defines) is never invoked. Clicks silently fail.	Add {{template "peer_detail_area" .}} after peers table
4	templates/route_table.html:18	Logic bug	high	hx-target="next tr" hx-swap="afterend" inserts detail after the NEXT row, not the clicked row. Broken for all but the last row.	Change to hx-target="this" hx-swap="afterend"
5	templates/peer_routes.html:25	Logic bug	high	Same hx-target="next tr" issue as #4	Same fix
6	handler_ui.go:116-181	Test gap	high	handleUISearch default/prefix branch has zero test coverage	Write test with type=prefix and empty type
7	render.go:49	Test gap	high	isBest template function has no test at all	Write TestIsBestTemplateFunc
8	handler_graph.go:60	Test gap	high	decorateGraphNodes never tested with active decorator; node names never verified	Write test with decorator that returns names
9	docs/guide/looking-glass.md:43-44	Stale doc	high	Lists removed routes /lg/search/aspath and /lg/search/community	Update to unified /lg/search
10	docs/architecture/web-interface.md:183-184	Stale doc	high	Same stale routes	Same fix
11	docs/architecture/web-interface.md:164	Stale doc	high	Lists deleted assets.go, missing new embed.go	Replace row
12	plan/learned/488-lg-looking-glass.md:23	Stale doc	high	Says "HTMX JS shim is minimal" -- now replaced with real htmx.min.js	Update or remove
13	docs/comparison.md:150	Missing update	high	Looking glass row says "No" for Ze -- Ze has a looking glass	Change to "Yes"

Medium Findings

#	File:Line	Category	Severity	Finding	Fix
14	handler_ui.go:116-181	Data flow	medium	Family selector rendered in UI but never read by handleUISearch -- dead input	Read r.FormValue("family") or remove dropdown
15	handler_ui.go:138	Security	medium	AS path pattern allows spaces; fmt.Sprintf("rib show aspath %s") could inject extra command tokens	Consider structured command dispatch or token filtering
16	server.go:155	Security	medium	No WriteTimeout on HTTP server -- slow-reading clients hold goroutines	Add WriteTimeout (with SSE-aware override)
17	server.go:181	Error handling	medium	registerRoutes swallows fs.Sub error -- server has no routes but reports no error	Return error from registerRoutes
18	handler_ui.go:213	Error handling	medium	Peer routes page shows empty table with no error message when engine returns error	Add "Error": engineError(zeData) to data
19	handler_ui.go:99	Test gap	medium	handleUISearchForm (GET /lg/search) not tested, not in routing table	Add to routing tests
20	render.go:56	Test gap	medium	Custom eq function untested. Also shadows Go's builtin eq unnecessarily	Remove custom eq or test it
21	handler_ui.go:411	Test gap	medium	Peer IP sort order with multiple peers not tested	Write sort order test
22	docs/guide/looking-glass.md:38-45	Stale doc	medium	Describes old multi-page layout, not new tab layout	Update UI description
23	plan/spec-lg-overhaul.md:201	Missing doc	medium	Wiring test references lg-ui-peers.ci and lg-ui-graph.ci that don't exist	Create .ci files or update spec
24	plan/spec-lg-overhaul.md:294	Missing doc	medium	Files to Create missing templates/error.html	Add to list

Low Findings & Notes

#	File:Line	Category	Finding
25	render.go:80	Logic	formatNumCommas returns "0" for unrecognized types instead of string repr
26	render.go:56	Logic	Custom eq treats all non-string types as equal (nil == 42 == true)
27	handler_ui.go:58,117	Security	No MaxBytesReader on POST forms (10MB default)
28	handler_ui.go:274	Error handling	Route detail silently falls back to different peer's route
29	graph.go:111	Error handling	String-encoded ASN values silently dropped from AS path
30	handler_api.go:247	Error handling	writeJSON can produce partial JSON on encode failure (no buffer)

/ze-review running nine agents and consolidating findings by severity

From AI Process to Product Design

The CLI is the API

→ Every CLI command is automatically available to AI and programs

→ No separate API to learn: one interface for humans and machines

→ MCP transport: any AI assistant connects, gets full daemon control

→ REST + Swagger UI, gRPC with proto definitions, gNMI for YANG config

Self-describing runtime

→ `ze help --ai [--json]`: machine-readable reference from the live binary

→ `ze schema methods` / `ze schema events`: all RPCs and notifications

→ `ze skills get <name>`: version-matched knowledge in the binary

→ `ze doctor [--json]`: preflight checks before an agent acts

Structured diagnostics

→ `ze config validate --json`: stable codes, source spans, expected vs actual

→ `ze explain <code>`: look up any error programmatically, with examples

→ `ze config fix --plan --json`: repair candidates with safety labels — never edits files itself

The Plugin Architecture

Minimal engine

- Ze is a content-agnostic event bus: components connect to it
- BGP, L2TP, IKE, firewall, FIB, web and CLI are all components or plugins
- Fast-path direct calls exist where performance needs them

Self-contained plugins

- **47 plugins** today, each self-contained with its own YANG schemas
- Plugins register via Go `init()`: anyone can add or remove modules

```
Ze Engine Core  (event bus)
  ├─ BGP Component   (FSM, wire, reactor)
  ├─ L2TP Component  (tunnel FSM, PPP, PPPoE)
  ├─ IKE Component   (IKEv2 engine, Child SA)
  └─ Plugin Infrastructure
       ├─ bgp-rib    route storage + best-path
       ├─ bgp-rs     route server, RFC 7947
       ├─ bgp-rpki   origin validation, RFC 6811
       ├─ bgp-bmp    monitoring, RFC 7854
       ├─ firewall   nftables + VPP ACL
       ├─ fib        kernel + VPP dataplane
       └─ …
```

One YANG Schema Drives Everything

Plugin-owned schema

→ The global model is assembled from the plugins compiled into the binary

→ **2,469 config nodes** across 224 YANG schemas — not one monolith

→ Remove a plugin and its config, CLI, web UI, API, MCP tools, validation and docs all disappear with it

Same model, many interfaces

→ CLI tab completion

→ Web UI forms and navigation

→ Config validation

→ REST / gRPC API types

→ MCP tool parameters

→ Schema discovery

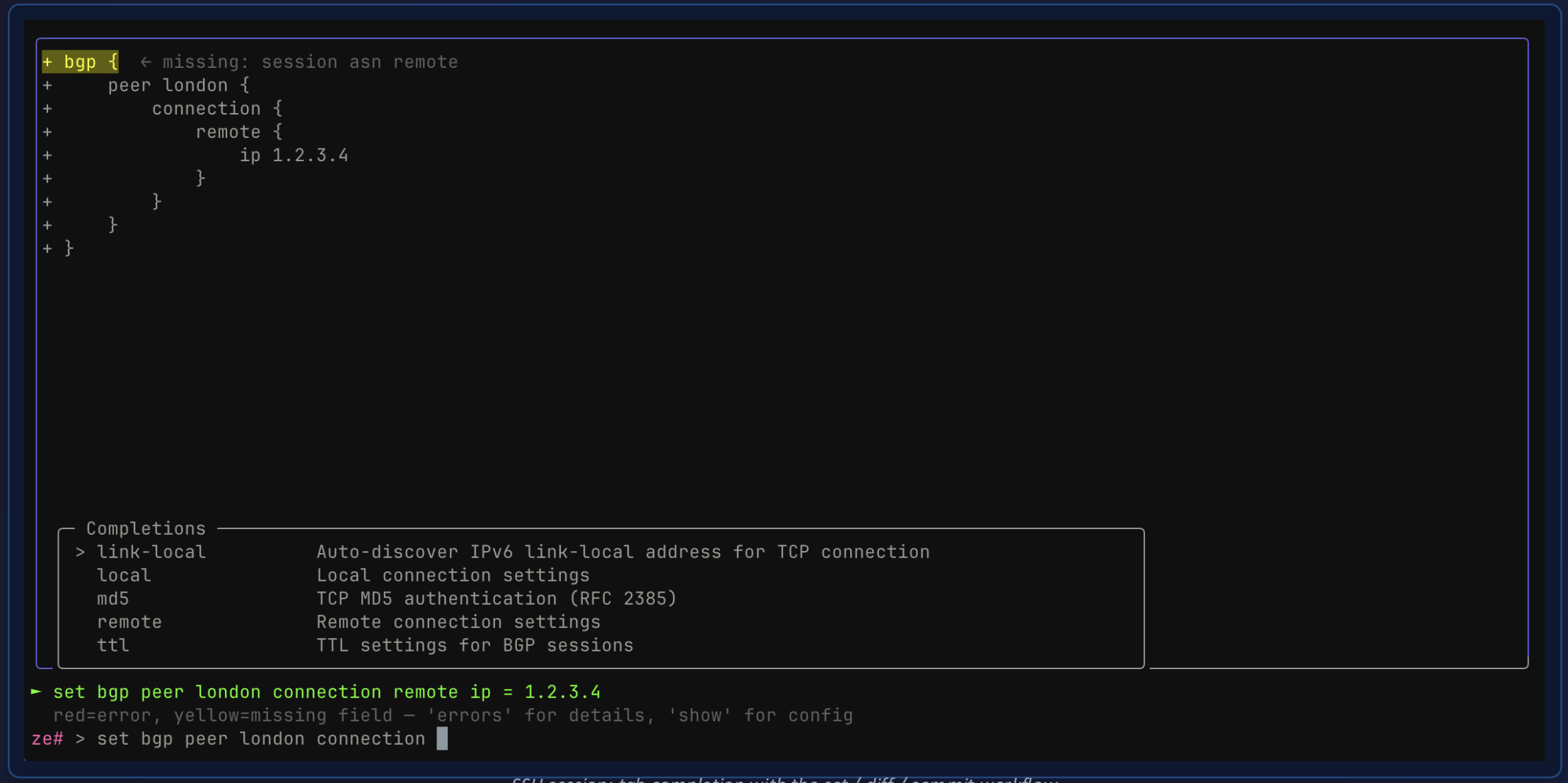
Network OS behavior

→ ZeFS stores config revisions with integrity checks and rollback

→ Config transactions, `commit confirmed`, reverse-tier rollback

→ Hot reconfiguration with automatic reconciliation

CLI



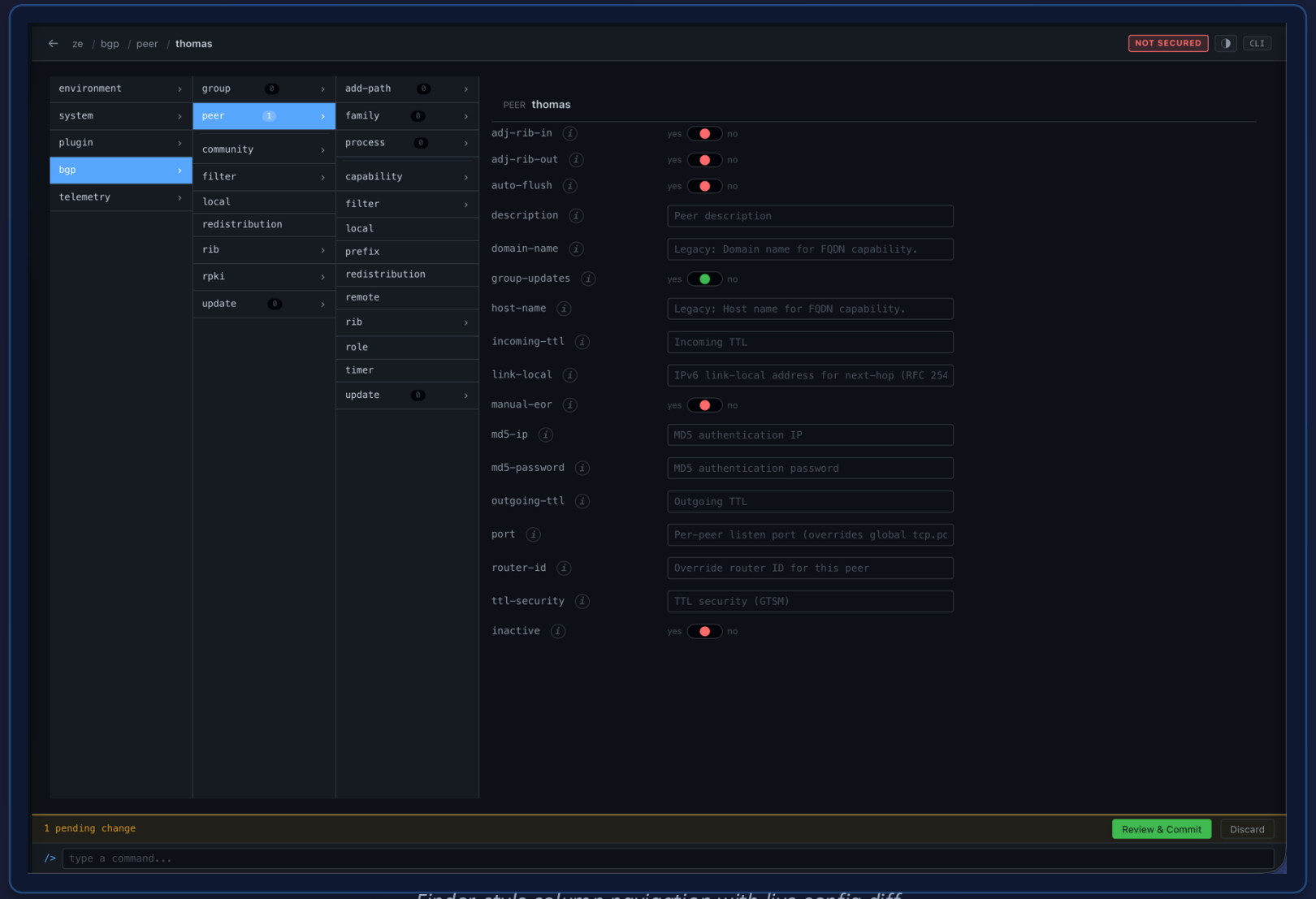
SSH session: tab completion with the set / diff / commit workflow

Web Interface

YANG-driven UI

- HTTPS server with **macOS Finder-style column navigation**
- Every UI element is generated from YANG schemas — zero hardcoded forms
- Per-user draft sessions with inline diff review and conflict detection
- Tab completion and live SSE updates when another user commits
- YANG decorators: AS numbers annotated with org names via Team Cymru DNS

Web Interface



Finder-style column navigation with live config diff

BGP Support

Protocol coverage

→ 21 address families: IPv4/6, VPN, FlowSpec, EVPN, VPLS, BGP-LS, MUP, MVPN

→ 13 capabilities: Add-Path, Extended Next-Hop, GR, LLGR, Roles, RPKI, ASPA, Hostname, Software Version

→ Full RFC 4271 best-path selection

Route server support

→ **RFC 7947 route server** is a plugin: one import → transparent route distribution

→ No FIB: a route server never forwards, so Ze skips kernel route installation

→ Interop-tested as a route server with FRR and BIRD as clients

→ BMP (RFC 7854): wire format, receiver, sender, Adj-RIB-Out (RFC 8671)

Built-in validation & ops

→ RPKI origin validation integrated into best-path selection

→ ASPA path verification

→ PeeringDB prefix-maximum updates

→ Public looking glass: route, AS path, community search with live SSE

Looking Glass

Peers | Search

Ze Looking Glass

BGP Peers

PEER	REMOTE AS	STATE	UPTIME	RECEIVED	ACCEPTED	SENT	UPDATES IN	UPDATES OUT	DESCRIPTION
127.0.0.2	65000	Established	54s				1,252	1,211,919	
127.0.0.3	65113	Established	54s				2,253	2,003,129	
127.0.0.4	65388	Established	54s				2,253	2,003,129	
127.0.0.5	64651	Established	54s				2,210,275	5,634	

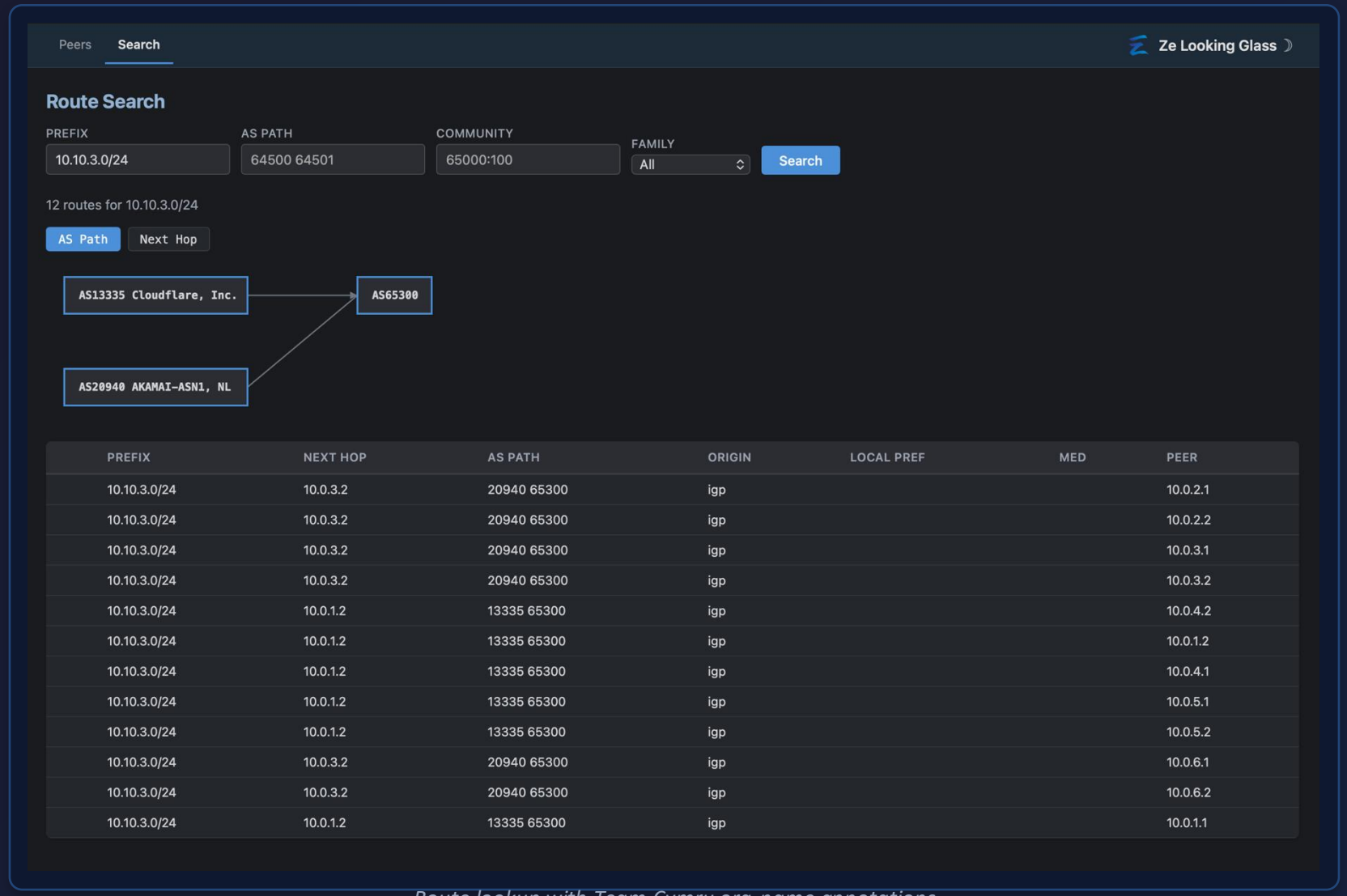
Routes from 127.0.0.5

Established AS 64651 Download CSV

FAMILY	PREFIX LENGTH	COUNT
ipv4/unicast	/24	1,000,000
ipv6/unicast	/54	858,493

Peer dashboard with AS-path topology

Looking Glass — Route Search



Route lookup with Team Cymru org-name annotations

Beyond BGP: Full Stack

IPsec / IKEv2

→ Native Go IKEv2 engine: FSM, crypto library, wire codec

→ EAP authentication, NAT-T, virtual IP pool, PKI certificate store

→ Interop-tested against strongSwan

L2TP + BNG

→ Complete L2TPv2 / PPP stack for broadband network gateways

→ PPP: LCP, IPCP, IPv6CP, PAP, CHAP-MD5, MS-CHAPv2

→ PPPoE access concentrator, RADIUS, DHCPv6-PD

Firewall, TC & VPP

→ nftables backend with expression lowering

→ FlowSpec firewall: BGP FlowSpec rules translated to nftables

→ VPP ACL backend, NAT44-ED, FIB programming, stats telemetry

→ Same config, plugins and CLI — the backend is a plugin choice

Performance

Two use cases pull opposite ways

→ **Route announcement** (ExaBGP case): optimise for zero-copy generation and sending

→ **Router** (route server, RIB, filtering): must parse to filter and apply backpressure

High-performance architecture

→ Lazy-parsed WireUpdate, with update groups for peers sharing capabilities

→ ContextID forwarding: same encoding context → forward raw bytes

→ Buffer-first encoding into pooled bounded buffers; per-attribute-type pools with dedup

There are three kinds of lies: lies, damned lies, and benchmarks.

DUT	Convergence	Throughput r/s	p50	p99	Lost
bird	44 ms	2,272,727	16 ms	28 ms	0
ze	**71 ms**	**1,408,450**	**25 ms**	**54 ms**	**0**
openbgpd	472 ms	211,864	217 ms	461 ms	0
frr	537 ms	186,219	412 ms	532 ms	0

Testing Infrastructure

Coverage

→ **1,053 functional tests** (.ci): real config, real daemon, real wire output

→ 42 interop scenarios against 7 implementations in Docker: FRR, BIRD, GoBGP, OpenBGPd, RustyBGP, rustbgpd, FreeRTR

→ Fuzz testing on all wire parsers

Specialized testing

→ Chaos testing framework with web dashboard: convergence tracking and property verification

→ Editor tests (.et) for headless TUI simulation

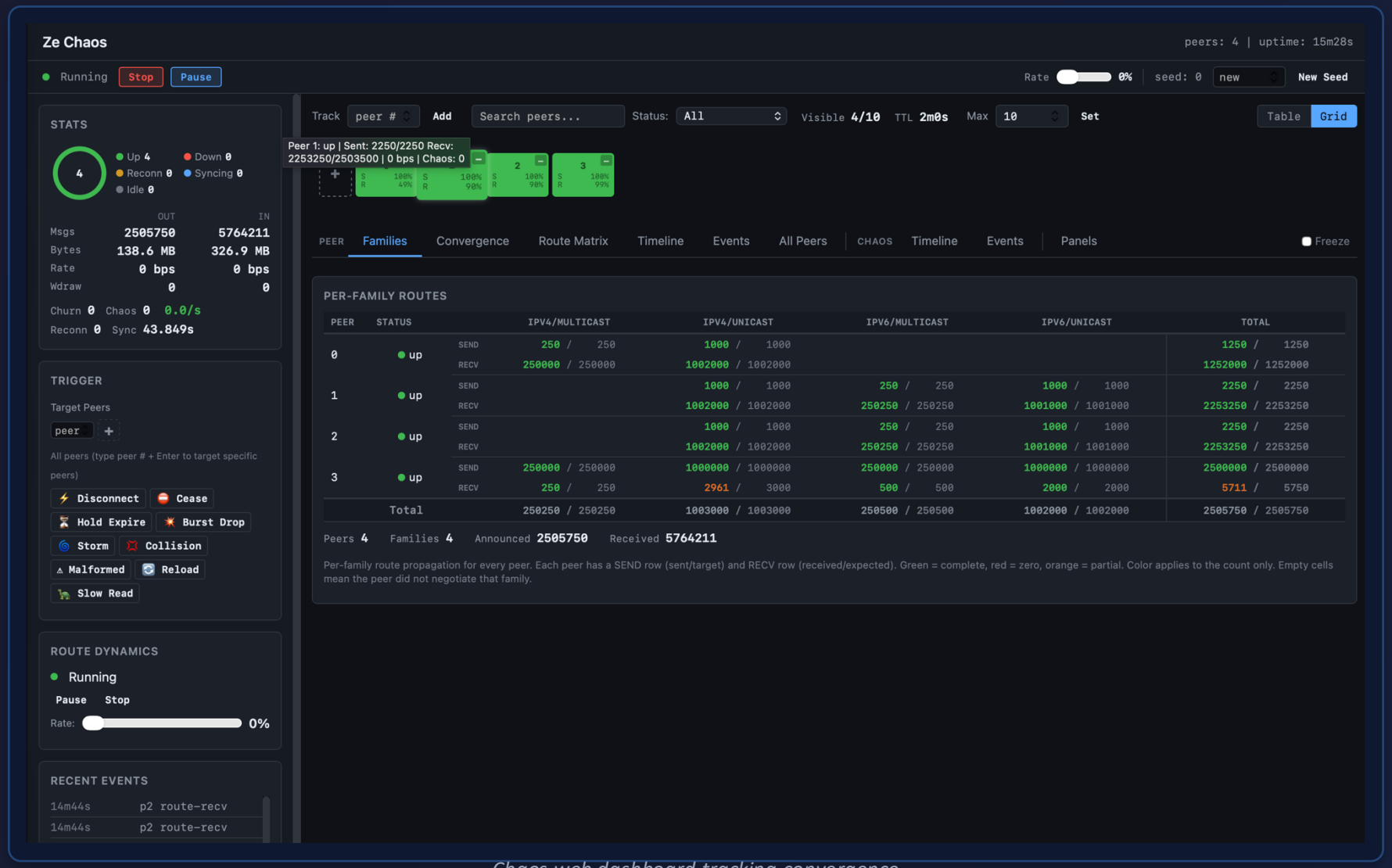
→ ExaBGP compatibility test suite

Gate

→ `make ze-verify`: 28 linters + unit + functional + ExaBGP tests

→ Full gate takes over 15 minutes — day-to-day work uses targeted checks first

Testing Infrastructure



Chaos web dashboard tracking convergence

ExaBGP Compatibility: Your Scripts Still Work

Migration tools

- `ze config migrate` converts ExaBGP configs to ze format automatically
- `ze exabgp plugin` runs existing ExaBGP processes with ze as the engine
- Bidirectional translation:
  - ze JSON ↔ ExaBGP JSON
  - ExaBGP commands ↔ ze commands

The promise

- Your existing scripts keep working: **upgrade the engine, not your tooling**
- Compatibility is tested; production mileage is still zero
- You can play with it now — but I will not pretend it has field history

Getting Started

Start here

- → Docs: **github.com/ze-software/ze/wiki**
- → Source & issues: **github.com/ze-software/ze**
- → Build from source: `make build`, then `bin/ze init`
- → Validate first: `bin/ze config validate <file>`

Deployment choices

- → Single static binary, cross-compiles to Go-supported Linux arches
- → Standard Linux daemon: `ze service install`
- → **Appliance image** with gokrazy (x86_64 default, arm64 supported)
- → Install media: **PXE provisioning** and **appliance ISO**

Help & debug

- → `bin/ze help command [filter]`: live command catalog
- → `bin/ze help --ai`: machine-readable reference for agents
- → `bin/ze --plugins`, `bin/ze schema list`: what this binary can do
- → `bin/ze -d <config>` or `ze debug enable <subsystem|all>`

Example: Route Server

```
plugin {
    internal rs { use bgp-rs; }
}
bgp {
    router-id 192.0.2.254;
    session { asn { local 65500; } }
    group ix-peers {
        connection {
            remote {
                ip dynamic;
                connect false;
                range 192.0.2.0/24;
                max-peers 200;
            }
            local { ip 192.0.2.254; accept true; }
        }
        session {
            rs-client true;
            next-hop unchanged;
            family { ipv4/unicast { prefix { maximum 10000; } } }
        }
        behavior { rs-fast-path enable; }
    }
}
```

Status

Current status

→ Exa Networks plans to run it — LINX lands a few weeks before that cutover

→ Lab + interop tested: ExaBGP compat, route-server, RPKI, BMP, VPP, IPsec, L2TP

→ No production deployment — **TESTING is next!**

→ Early adopters: treat as a controlled trial, it is not finished (UI in particular)

Since NetMcr, April 9th

→ IPsec, L2TP, firewall, VPP, REST/gRPC, gNMI, BMP, policy and config transactions are all native

→ The dev system compounds: patterns, specs, reviews and 881 summaries make the next feature easier

→ Only **871k lines** of Go code

→ Only **44M** of vendored code

Release position

→ **Pre-release:** looking for early adopters and feedback

→ **Careful release:** the YANG model *is* the API and not stable yet — it will change

→ Battle testing is the next hard part

Questions?

Now, today

- → Happy to take as much time as I have to answer

Later

- → Point your AI at the repo and ask it
- → It can answer on my behalf — probably more accurately

Happy to help

- → Discuss Ze, ask for a change, report something wrong, or influence where it goes — talk to me
- → Contributions need not be code: ideas, questions, operational feedback and weird deployment scenarios all help

Discord `discord.gg/ykJb8meS4`

Repo `github.com/ze-software/ze`