THE EVIDENCE

A Structured Record of Construction, Cadence, and Coordination

Companion to "The System of Work."

Snapshot: 2026-05-07. HEAD `cd440c45...` *of* `Eigenly-Inc/fireweed-platform`*. All measurements reproducible from the commands in Appendix A.*

> *The thesis names the layer. The sections below record what it looks like in one repository at one snapshot. Numbers are observations, not arguments. The reader draws the conclusions.*

Setup

Eigenly is a pre-seed startup. The construction below was produced under real economic and time pressure, not research conditions. Christian and Vaibhav — two engineers — built the company, the Fireweed platform, the Wishflower commerce product, and the supporting infrastructure from zero in parallel, with a deliberate posture of production-grade-from-day-one rather than build-now-refactor-later. The work spans roughly six months of calendar time, which resolves to 110 active mainline-commit days (see §3.2).

The two engineers worked alongside four AI tools used in parallel based on what each is strongest at: OpenAI Codex for planning and implementation, Claude Code for out-of-the-box thinking, creativity, and prototyping, Gemini CLI for data augmentation, and Replit for fast design exploration. The tool mix was pragmatic and concurrent — different tools for different work shapes, often in the same hour — not a sequential pipeline. None of the four tools was the system; the System of Work substrate this document records is what they all coordinated against.

---

0. Why This Was Built

Every organization that ships anything coordinates through a substrate. Sprint boards, document templates, approval chains, on-call rotations, naming conventions, shared assumptions about who decides what, oral history about why a particular decision was made. Most of that substrate is implicit — carried in the heads of long-tenured people, propagated through proximity, repaired continuously by the social work of asking, escalating, double-checking, summarizing.

Implicit substrates work as long as the actors share enough context to do that repair quietly.

Small teams do it through proximity. Larger ones do it through tenure. Both depend on a cadence at which humans can absorb, infer, and propagate the missing structure faster than work creates new gaps.

AI agents do not perform that repair. They do not infer authority from social context. They do not know what was decided last quarter unless it is encoded somewhere they can read. When an agent's context window closes, anything not written down is gone. When several agents work in parallel against the same code, conventions that lived as social etiquette become merge conflicts. When agents operate at machine cadence, the gap between what is true and what is written down widens faster than any human team can close it through conventional documentation work.

The hypothesis behind this construction is straightforward: if AI is going to be a first-class participant in work, the substrate that lets actors coordinate has to be first-class too. Not adjacent to the code, not maintained out of band, not carried in social memory. Encoded as inspectable, queryable, governable artifacts that humans and agents read from the same source of truth.

This document records the construction that resulted from running that hypothesis for roughly six months, with two humans and many AI threads operating in parallel against a single repository. What was built, at what cadence, with what coordination pattern, under what governance, and what is currently live in production. The thesis frames why this layer matters; the sections below show what it looks like in this repository at this snapshot.

---

1. What Was Built

The repository spans implementation, contract surface, infrastructure, deployment estate, and test footprint. Numbers below are line-counted with `wc -l` against `HEAD=cd440c45...` and reproducible from Appendix A.

1.1 Implementation surface

633,406 non-generated implementation lines across Rust services and CLI tooling, TypeScript services and frontends, and Python tooling. With generated TypeScript contract stubs included, the total is 689,374 lines. Rust dominates the backend lane (562 files / 404,104 lines), TypeScript the surface and orchestration lane (573 files / 206,809 lines), TSX the frontend lane (215 files / 57,951 lines).

1.2 Service and product surface

Sixteen backend services under `services/` (fifteen Rust, one TypeScript). Five apps under `apps/`: `web` (operator surface), `wishflower` (commerce surface), `wishflower-native` and `fireweed-native` (mobile), `eigenly-site` (public/marketing). Eighteen shared packages

under `packages/` and six Rust crates under `crates/`. The operator/control-plane CLI under `tools/fireweed/src/` is itself 237 Rust files / 279,656 lines — larger than any individual backend service, with `main.rs` alone at 23,920 lines. It is the binary that executes the lifecycle logic invoked from both human terminals and CI runners (see §2.8).

1.3 Contract surface

Thirty `.proto` files in `contracts/proto` across twenty-six top-level domains: `agent`, `asset`, `attention`, `auth`, `catalog`, `chat`, `codex_bridge`, `common`, `context`, `culture`, `decision_queue`, `delegation`, `injection`, `memory`, `presence`, `room`, `room_bridge`, `room_topology`, `search`, `search_evals`, `surface`, `surface_feed`, `tool`, `work`, `work_intelligence`, `workflow`. Four domains carry multi-version surfaces (`work`, `surface`, `room`, `presence`). Generated TypeScript stubs: 61 files / 55,968 lines, regenerated and drift-checked in CI.

1.4 Infrastructure surface

Eighty-seven forward-only Postgres migrations from `0001__roles.up.sql` through `0087__replay_pack_service_app_grants.up.sql`, with row-level security enabled across rooms, surfaces, work kernel, delivery graph, auth plane, agent runtime, memory plane, workspace connectors, and Codex bridge tables. Nineteen Helm charts (sixteen service + one migration + one base + one sample). Twenty-six GitHub Actions workflows. Pulumi-managed GCP data plane (Cloud SQL Postgres 18 private, three GCS buckets, BigQuery dataset, Redis, eighteen service accounts, sixteen runtime Secret Manager secrets) and Cloud DNS IaC for `eigenly.com` and `eigenly.ai`.

1.5 Live deployment

As of 2026-05-07, public health endpoints respond:

URL	HTTP	Time
`https://api.eigenly.com/health`	200	0.185s
`https://assets.eigenly.com/health`	200	0.273s
`https://eigenly.com`	200	0.533s
`https://fireweed.eigenly.com`	200	0.329s
`https://wishflower.eigenly.com`	200	0.379s

DNS resolution is live across `api`, `ws`, `assets`, `fireweed`, `wishflower`, and the apex `eigenly.com` and `eigenly.ai` zones, backed by a mix of GKE gateway and Firebase

Hosting.

1.6 Test footprint

Three hundred thirty-nine broad test files / 75,726 test lines under the standard project test conventions. 2,667 measured Rust and TypeScript test functions (1,723 Rust `#[test]` / `#[tokio::test]` attributes + 944 TS `it` / `test` calls). 371 Bazel query test targets and 223 BUILD-file test macros (110 `sh_test`, 59 `rust_test`, 53 `py_test`, 1 `buildifier_test`). The E2E suite under `tests/e2e/` includes 18 Playwright spec files, 27 shell scripts, and 23 Bazel `sh_test` E2E targets, with deployed-system smoke proofs for hosted auth, Wishflower room, Fireweed observation, execution atlas, search evals, work intelligence, and workspace connectors.

---

2. The Substrate as Artifact

The thesis names seven primitives the coordination substrate decomposes into: identity, context, memory, coordination, intent, governance, provenance. This section maps those primitives to artifacts in the repository. The mapping is not strict — some artifacts span multiple primitives, and the boundaries are continuous rather than discrete.

2.1 Identity

Seventeen formal domain agents are defined under `.claude/agents/*.md` (1,015 lines) with parallel `.codex/agents/*.toml` projections (877 lines). Seventeen System-of-Work domains under `.work/system-of-work/domains/` carry their own `context/`, `policy/`, and `owners` declarations. Agent and human participants have stable, inspectable definitions — capability boundaries, posture, and scope carried in files rather than in convention.

2.2 Context

The `.work/system-of-work/` corpus (679 markdown files / 53,520 lines + 77 structured files / 19,847 lines) holds the active operating contract. `ROUTER.md` (524 lines) is the non-negotiable workflow entrypoint. `EXECUTION_CONTRACT.md` (128 lines) defines what "correct" means at runtime. `FOUNDATIONS.md` (682 lines) carries operating doctrine. `DECISION_CAPTURE_CONTRACT.md` (106 lines) defines how decisions must be captured. Schemas, registries, and the build graph make context queryable rather than narrative.

2.3 Memory

Two hundred fifty-six active and done task and effort packets under `.work/tasks/` (120 in progress + 136 done) plus sixty-one exploration packets under `.work/exploration/` (40 in progress + 21 done). Each packet carries scope, intent, plans, verification, learning, and provenance. The done corpus alone holds 2,174 markdown files / 348,464 lines — averaging

roughly 2,562 markdown lines per closed packet directory. Total `.work/` retained corpus: 1,838,449 line-counted files (901,532 markdown + 371,967 structured + 564,950 other retained evidence).

2.4 Coordination

The Room model — three rails (action, event, presence) — provides the runtime coordination primitive. Service boundaries are governed by the thirty `.proto` files described in §1.3, with drift enforcement in CI. Procedure-level coordination — 127 workflow specs and 73 governed slash-command skills that expose them as executable surfaces — is the subject of §2.8.

2.5 Intent

One hundred thirty-five effort briefs under `.work/tasks/*/effort-brief.md` carry per-effort intent. Two hundred nineteen product-intent markdown files under `docs/product-intent/` (23,046 lines) decompose into product decision records (22), evaluation records (12), experience specs (20), feature records (33), interaction vocabulary (42), room compositions (6), Fireweed-specific intent (12), and Wishflower-specific intent (63). One hundred fifty-nine ADRs ratify architectural intent at the platform level.

2.6 Governance

One hundred fifty active ADRs and nine archived ADRs (159 total) carry architectural authority, with 25,902 references across the repository. Supersession discipline is visible in archived chains (ADR-0004 → ADR-0046; ADR-0071 → ADR-0087 → ADR-0134; ADR-0120 → ADR-0164). Ten machine-readable invariants under `.work/decision-records/invariant-registry.json`, all currently in advisory rollout mode, define platform-level constraints that workflows and CI gates can ratchet against.

2.7 Provenance

Eight hundred fifty-eight verification reports under packet `verification/` directories. Four hundred twenty-two learning package files. Sixteen thousand two hundred reachable commits through 2026-05-06 with full human-author attribution. Five hundred thirty-six co-author trailers carry explicit AI/tool attribution in commit metadata (483 model-or-tool attributions across 128 generic Claude, 108 Claude Opus 4.7 1M, 103 Claude Opus 4.6, plus Codex and Cursor entries). The trailer-level commit attribution discipline began partway through the snapshot window; for commits prior to that, AI participation is reconstructed by the work-intelligence layer through intersecting local Codex and Claude Code session logs with commit timestamps and changed-path sets. Local Codex JSONL session logs hold 5.47 million response items and 1.96 million tool-call records with timestamps; local Claude Code transcripts hold 24,014 tool-use records. Provenance is structural — encoded in commit metadata where the trailer discipline applied, reconstructable from session logs where it did

not.

2.8 The substrate compiles

The primitives above are not stored only as prose. There is an explicit projection chain from procedure spec → governed slash-command surface → projected agent skill surface → compiled control-plane binary → CI invocation, with humans, agents, and CI calling the same binary as the source of execution truth. The dependency map is declared in `.work/system-of-work/POLICY.md`.

Workflow specs under `.work/system-of-work/domains/{domain}/workflows/{name}.md` — 127 procedure sources — carry frontmatter declaring triggers, executors, commands, and related files. Of those 127, 73 have a corresponding governed slash-command skill at `.claude/skills/{command}/SKILL.md`; all 73 carry the required `governed-by`, `router-entry`, and `update-workflow` metadata, and all 73 link back to the workflow spec that governs them. The other 54 workflow specs are governed procedure source without slash-command adapters — typically lower-level or non-interactive procedures. Each Claude-side skill projects to a corresponding `.agents/skills/{command}/SKILL.md` agent-facing surface (73 projections). Domain agent definitions follow the same pattern: 17 Claude agents under `.claude/agents/*.md` project to 17 Codex TOML projections under `.codex/agents/*.toml`. The build graph at `.work/system-of-work/build-graph/` (8 files / 15,614 lines / 618 command entries) makes governance dependencies and validation commands machine-readable.

The compiled control plane lives at `tools/fireweed/src/` — 237 Rust files / 279,656 lines. Load-bearing subsystems include `sow/` (23 files / 24,004 lines, validates workflows, skills, routers, projections, build graph, reports, signals, sessions), `tasks/` (47 files / 39,923 lines, implements start-work / verify-work / update-pr / finalize-pr / ready / merge / repair / integration flows), and `ci*` (14 files / 14,210 lines, owns CI selection, release maps, manifests, promotion/rollout diagnostics). Forty-two `sh_test` Bazel targets under `tools/scripts/` provide thin Bazel-addressable guardrail checks. CI invocation matches the same coupling: of 26 GitHub Actions workflows, 17 call `bazelisk run` or `bazel run` (93 occurrences) and 17 invoke repo-owned `//tools/*` Bazel labels (101 occurrences). The `//tools/fireweed:fireweed_ci` label appears in 6 workflows / 31 occurrences; `//tools/fireweed:fireweed_ready_guard` appears in 1 workflow / 4 occurrences. CI does not reimplement procedure logic in YAML; it calls the same binaries humans and agents call.

End-to-end worked example: the `verify-work` procedure is sourced at `.work/system-of-work/domains/tasks/workflows/verify-work.md`, projected into `.claude/skills/verify-work/SKILL.md` (frontmatter: `governed-by: .../verify-work.md`), executed via `bazelisk run //tools/fireweed:fireweed -- tasks verify-work ...`, and gated in CI by `.github/workflows/verification-report-guard.yml` invoking

`//tools/fireweed:fireweed_ready_guard`. Same binary, same lifecycle logic, same source of truth, three invocation paths.

---

3. The Cadence

Six months, two humans, parallel AI threads. The repository activity profile follows.

3.1 Volume

Sixteen thousand two hundred unique reachable commits through 2026-05-06. One thousand four hundred eleven merged GitHub PRs through the same date window (1,413 in a later live rerun, after two post-HEAD merges). Four thousand four hundred thirty-nine commits in the last thirty days; 10,339 in the last sixty; 15,547 in the last ninety. Authorship across all refs: Christian Blank 15,031 commits (92.78%), Vaibhav Gumashta 1,049 commits (6.48%), Claude 76 commits (0.47%), Fireweed Cloud Codex 33 commits (0.20%), mergify 11 commits (0.07%). Two human operators committed against the same governance substrate; AI participation is encoded in commit metadata where the trailer discipline applied and reconstructed from local session logs through the work-intelligence layer where it did not (see §2.7).

3.2 Mainline compression

The `origin/main` ref carries 1,670 commits across a 194-day calendar span (2025-10-26 → 2026-05-07). Of those 194 calendar days, only 110 had at least one mainline commit; 84 had none. The 110 active days break down as 81 weekdays and 29 weekends. The largest stretch with zero mainline commits was 52 days (2025-12-05 → 2026-01-25), 36 of which were weekdays. Active days saw an average of 15.18 mainline commits per day; the calendar-day average was 8.61. Peak: 72 mainline commits on 2026-03-04.

This is not six continuous months of even work. It is concentrated bursts of mainline landings interspersed with branch and exploration work that does not appear on `origin/main`. The total reachable commit volume (16,200) is roughly an order of magnitude larger than the mainline landing volume (1,670), reflecting the branch-and-rebase cadence of multi-thread agent-assisted work.

3.3 Concurrent commit signature

Across all refs, there are 2,473 distinct minutes with two or more commits and 38 distinct minutes with six or more commits. Peak minute: 25 commits at 2026-04-15T23:28. Peak day across all refs: 1,018 commits on 2026-03-04. This pattern is not consistent with single-keyboard serial work; it is consistent with multi-branch, multi-thread orchestration.

3.4 Time-of-day inversion

Commits across the last sixty days, grouped by author hour, peak from late evening through early morning. The four highest-volume hours are 00:00 (687), 01:00 (650), 02:00 (642), and 23:00 (611); the four lowest are 10:00 (202), 11:00 (212), 09:00 (241), and 08:00 (270). The shape is consistent with an operator orchestrating asynchronous AI work across time-zones rather than coding in office-hour bursts.

---

4. The Coordination Substrate in Operation

This section reports local-machine telemetry from the Codex Desktop and Claude Code clients on the operator's machine. Unlike the rest of the document, it does not derive from repo-committed evidence; snapshot SHAs and reproduction scripts are in Appendix A.

4.1 Codex thread substrate

One thousand six hundred twenty-nine JSONL session files (28.31 GiB, zero parse errors) were captured between 2025-11-03 and the 2026-05-07 snapshot cutoff. The Codex Desktop SQLite snapshot recorded 1,571 threads (525 root + 1,046 spawned child threads), with 187 billion `tokens_used` recorded across the full corpus and 27 individual threads above 1 billion tokens.

The session corpus contains 1,963,516 actual tool-call records: 1,074,427 `exec_command`, 742,616 `write_stdin`, 83,306 `apply_patch`, 15,052 `update_plan`, 10,424 `spawn_agent`. The work was operational execution, not chat.

4.2 Subagent fanout

One hundred forty-seven parent threads spawned children. Maximum fanout from a single parent: 50 children. Thirty-one parents spawned ten or more children; sixty-five spawned six or more. The role mix maps to System-of-Work domains rather than to generic chat labels: explorer 448 child threads, tasks 124, infra 94, system 87, auth 50, worker 33, devex 19, bazel 25, ADR 25, plus smaller counts across vision, UI, contracts, product-intent, agents, collaboration, commerce, and work.

4.3 Bounded autonomy with explicit escalation

Half of all Codex sessions (50.28%) ran from a single coalesced input boundary. For spawned subagent sessions specifically, that figure rises to 64.20% — subagents typically receive one input and execute through to completion. Subagent autonomous-window p90 is 83.29 minutes; p95 is 179.15 minutes. Top-level operator threads are different: longer wall spans, more interventions, more orchestration. The pattern is operator threads coordinate; bounded subagents execute.

Explicit `request_user_input` calls appear in 28.30% of all sessions and have grown month-

over-month from zero in late 2025 to roughly one-third of sessions in April–May 2026. The growth reflects a migration of human input upward in the chain rather than a decline in autonomy. Early in the snapshot window, efforts were small implementation slices: a human picked a tiny scope and handed it to an agent. By the end of the window, efforts are large multi-phase atlas constructions where the human plans, co-designs, architects, and product-manages at the atlas level, then the work breaks down into per-phase efforts that AI implements largely autonomously. Effort 0160 — 100 files / 12,567 markdown lines across twelve phases — is the snapshot's clearest instance of this atlas pattern. Input requests now cluster at boundaries above and after implementation: atlas-level planning (which captures increasingly larger scope as the substrate matures), verification, and post-effort handoff into "what next." Inside implementation execution itself, input requests are nearly absent. The substrate's load-bearing function is to absorb increasingly large planned efforts autonomously while preserving the human-input boundaries where judgment is needed.

4.4 Long-running root work streams

Turn-lifecycle pairing of `task_started` and `task_complete` events in the SQLite snapshot shows root threads with substantial multi-turn runtimes: maximum single-turn duration 20:30:43, maximum thread active runtime 99:58:17 (sum of paired turns), maximum thread span 188:31:40. The five longest root threads each handled 17 to 76 user-message records across 17 to 73 paired turns, with tool-call counts between 12,180 and 21,330 per thread and `apply_patch` counts between 434 and 1,579. These are not chat sessions; they are multi-day work streams with continuous tool-driven progress and multiple human interventions throughout.

Aggregate paired-turn runtime across the snapshot, overlaps not subtracted: 115.24 days total across root and child threads.

4.5 Claude Code thread substrate

The Claude side of the operating model is captured in 646 transcript files (60 root + 586 subagent), 0.241 GiB, zero parse errors. One hundred parent sessions spawned Claude subagents; max fanout from one parent was 22 children. Tool-use records: 24,014. Top tools: `Bash` 9,409, `Read` 8,865, `Edit` 1,179, `Glob` 1,040, `Write` 824, `Grep` 694, `Agent` 275.

Effective input context tokens recorded in usage fields total 5.26 billion (input 3.5 million + cache-creation 247.7 million + cache-read 5.0 billion). Output tokens: 19.2 million. Model mix in transcripts: claude-haiku-4-5 16,314 records, claude-opus-4-7 11,043, claude-opus-4-6 6,637, claude-sonnet-4-6 799.

Root continuation chains (resumed sessions threaded across multiple wall-time gaps) reach a maximum of 250:23:51 wall span, with gap-capped active runtime of 06:10:02 inside

those long spans. The session-inventory wall-span maximum is 250:24:58. These long spans are not continuous-work claims; they are sessions that remained open and were resumed across multiple work windows. The active-runtime number is the defensible work-time proxy.

4.6 Concurrency

Codex event-active concurrency peaked at 14 distinct sessions emitting events in the same UTC minute, with 2,029 minutes showing six or more active sessions and 106 minutes showing ten or more. Claude Code event-active concurrency peaked at 8 active transcript files in one minute, with 779 minutes showing two or more and 40 minutes showing six or more. The two clients operated as a multi-thread substrate rather than as a serial assistant.

4.7 Tool neutrality

The telemetry in §4.1–§4.6 was generated through ordinary Codex Desktop, Codex CLI, Codex VS Code, and Claude Code clients running against the file-based substrate in this repository. No third-party agent orchestration platform, custom cloud AI harness, or bespoke OpenAI/Anthropic API integration drove the work from outside the repository. The AI clients executed against repository files; coordination state — context, packets, workflows, ADRs, invariants, verification reports — persisted in repository files. The intelligence layer and the coordination substrate were architecturally separate, communicating through standard developer tools (Git, GitHub, Bazel, Postgres) and the platform's own repository-internal Codex bridge service (`services/codex-bridge-gateway` , 8,158 lines, with explicit deletion triggers governing its bridge-tech status). The substrate continued to function across the model and reasoning-effort changes visible in the SQLite snapshot (gpt-5.2, gpt-5.3-codex, gpt-5.4, gpt-5.4-mini, gpt-5.5, claude-opus-4-6, claude-opus-4-7, claude-sonnet-4-6, claude-haiku-4-5) without modification to the substrate itself.

---

5. The Governance Loop

The construction in §1 and the cadence in §3 are sustained by enforcement, not by intent. Twenty-six GitHub Actions workflows, packet-lifecycle artifacts, ADR supersession discipline, and an invariant registry close the loop between work and governance.

5.1 CI gates

Load-bearing workflows include `CI` (Bazel `//:preflight` or `//:preflight_light` plus full `bazelisk build //...` and `bazelisk test`), `Proto Drift Guard` (regenerates and diffs TS proto stubs, checks shared types and exports, builds Rust contract baseline), `Verification Report Guard` (enforces committed verification report and canonical

comment posture per PR), `System of Work Validation` (runs `fireweed sow validate`), and `Governance Self-Audit` (runs `fireweed sow report governance` and opens issues when missing-design-review, missing-verification, missing-north-star-proxy, or expired-waiver counts are non-zero).

Across the full 26 GitHub Actions workflows, 17 call `bazelisk run` or `bazel run` (93 occurrences) and 17 invoke repo-owned `//tools/*` Bazel labels (101 occurrences). The CI/CLI coupling is structural: workflows orchestrate scheduling, credentials, and runners, while the procedure logic lives in compiled control-plane binaries (see §2.8) that humans and CI invoke against the same source of truth.

April 2026 GitHub Actions workflow-run counts via API: 22,726 across all workflows, of which 5,622 were Verification Report Guard runs, 3,859 were CI runs, 728 were SoW Validation runs, and 445 were Proto Drift Guard runs. CI is the dominant per-PR gate; verification governance fires multiple times per PR lifecycle.

5.2 Packet lifecycle

Two hundred fifty-six task and effort packets under `.work/tasks/in_progress/` and `.work/tasks/done/`. Per-packet artifact counts across the corpus: 741 phase plans (`phase-*.plan.md`), 858 verification reports under packet `verification/` directories, 422 learning package files, 532 design-review files, 135 effort briefs. Effort 0160 — the closeout effort current at this snapshot — alone contains 100 files / 12,567 markdown lines including phase plans 00 through 12, phase verification reports 00 through 12, learning packages 00 through 12, design reviews for phases 03/05/06/07/09/10/11/12, an execution atlas, and closeout artifacts (final evidence index, governance ratchet, cleanup closure, retrospective).

5.3 ADR supersession

ADRs are not append-only. Nine are formally archived under `docs/adr/archived/` with explicit supersession links. Examples: `ADR-0004` → `ADR-0046`, `ADR-0045` → `ADR-0028`, `ADR-0071` → `ADR-0087` → `ADR-0134`, `ADR-0120` → `ADR-0164`. Of the 159 ADRs, 25,902 in-repo references make ADR authority queryable rather than ceremonial. High-reference ADRs include ADR-0078 (1,268 references), ADR-0118 (854), ADR-0001 (776), ADR-0085 (756), ADR-0023 (590), ADR-0133 (571).

5.4 Invariants

Ten invariants in `.work/decision-records/invariant-registry.json`:

`rooms_are_interaction_primitive`, `products_are_room_centered_compositions`, `platform_recursion_invariant`, `kernel_capability_boundary`, `capability_composition_explicit_and_typed`, `participants_are_first_class_and_role_scoped`,

`adaptive_outputs_require_explicit_audience_scope`, `modality_is_adapter_not_protocol_fork`, `memory_and_personalization_require_provenance`, `incentives_must_stay_user_aligned`. All currently in advisory rollout mode. The mechanism for ratchet from advisory to blocking exists; no invariant is yet a hard commit-time wall.

---

6. The Live System

Construction is not deployment. This section records what is actually running in production at the snapshot date.

6.1 Public endpoints

Five HTTPS endpoints respond healthy as of 2026-05-07:

URL	Code	Time
`https://api.eigenly.com/health`	200	0.185s
`https://assets.eigenly.com/health`	200	0.273s
`https://eigenly.com`	200	0.533s
`https://fireweed.eigenly.com`	200	0.329s
`https://wishflower.eigenly.com`	200	0.379s

6.2 DNS and ingress

`eigenly.com`, `www.eigenly.com`, `eigenly.ai`, and `www.eigenly.ai` resolve to Firebase Hosting (`199.36.158.100`). `api.eigenly.com`, `ws.eigenly.com`, and `assets.eigenly.com` resolve to GKE gateway IP `34.58.163.198`. `fireweed.eigenly.com` and `wishflower.eigenly.com` resolve through Firebase Hosting CNAMEs (`fireweed-platform.web.app`, `wishflower-platform.web.app`). MX, SPF, and DMARC records on `eigenly.com` route through Google Workspace.

6.3 Infrastructure substrate

The `packages/infra-sdk/src/gcp/data_plane/pulumiProgram.ts` Pulumi program creates one Cloud SQL instance (`POSTGRES_18`, private VPC-only, point-in-time recovery), three Cloud Storage buckets (work-intelligence raw evidence; runtime assets; replay packs with 7-day lifecycle delete), one BigQuery dataset and one `facts_v1` table (DAY-partitioned on `event_date`, clustered by `tenant_id` / `work_key` / `work_session_id` / `commit_sha`), one Redis

instance, eighteen service accounts, sixteen runtime Secret Manager secrets, and SecretProviderClass Kubernetes resources for DB, Redis, OIDC, workspace connectors, OpenAI, Anthropic, GitHub governance bot token, asset URL signing, internal auth tokens, and agent runtime identity. Workload Identity bindings tie service accounts to Kubernetes service accounts.

6.4 Cluster deployment

Nineteen Helm charts under `infrastructure/kubernetes/charts/` : sixteen service charts (one per backend service), one `db-migrations` chart, one `fireweed-service` base chart, and one `hello-world` sample. Service charts include Workload Identity validation (`serviceAccount.gcpServiceAccountEmail` required), Cloud SQL proxy sidecars, health probes, autoscaling configuration, and HTTP/gRPC port routing. Eighty-seven forward-only Postgres migrations enable RLS across rooms, surfaces, work kernel, delivery graph, auth plane, agent runtime, memory plane, workspace connectors, and Codex bridge tables.

6.5 DNS as code

`infrastructure/domains/domains.dev.yaml` declares twenty-six logical DNS records across two zones: `eigenly.com` (21 records: Firebase Hosting, GKE gateway A records, ACME validation, Google Workspace MX/SPF/DKIM/DMARC) and `eigenly.ai` (5 records: Firebase Hosting + ACME). The `packages/infra-sdk/src/domains/pulumiProgram.ts` program creates the corresponding `gcp.dns.ManagedZone` and `gcp.dns.RecordSet` resources, with mapping/program/deploy/doctor tests under `packages/infra-sdk/tests/domains*.test.ts` .

---

7. Spend

The AI/tooling spend across the snapshot window is receipt-backed by user-provided email-derived screenshots and operator-confirmed for the cofounder add-on. The arithmetic is reproducible from `local-reports/ai-spend-ledger-2026-05-07.json` via `local-reports/scripts/ai_spend_audit.py` .

Bucket	Amount
Email-derived transactions (26 transcribed)	$2,579.12
Operator-confirmed cofounder add-on	$2,800.00
Same-window total	**$5,379.12**
Average per observed billing month (7 buckets)	$768.45

Vendor breakdown across the canonical same-window total:

Vendor	Email-derived	Cofounder add-on	Total
Anthropic / Claude	$1,445.00	$1,400.00	$2,845.00
OpenAI / Codex / API	$940.92	$1,400.00	$2,340.92
Replit	$193.20	$0.00	$193.20
Total	**$2,579.12**	**$2,800.00**	**$5,379.12**

Monthly buckets:

Month	Email-derived	Add-on	Total
2025-11	$250.00	$400.00	$650.00
2025-12	$225.00	$400.00	$625.00
2026-01	$225.00	$400.00	$625.00
2026-02	$281.15	$400.00	$681.15
2026-03	$516.29	$400.00	$916.29
2026-04	$831.68	$400.00	$1,231.68
2026-05 (through May 7)	$250.00	$400.00	$650.00

Inclusion notes: the 26 email-derived transactions cover Anthropic, OpenAI, and Replit charges across November 2025 through May 7, 2026. The cofounder add-on is operator-confirmed at $400/month (one Codex Pro subscription + one Claude Code subscription) modeled across the same seven billing-month buckets. Gemini CLI does not appear as a separate billable line in the email-derived records for this window. A separate $710.00 mobile screenshot reporting Anthropic and OpenAI charges was excluded from the canonical total because multiple detailed entries within it are dated March/April 2025, outside the claimed November 2025–May 2026 window; including it would risk importing out-of-window spend or double-counting overlapping subscription records. Raw receipts and screenshot binaries are not committed; the ledger holds only transaction-level amounts, dates, vendors, and source labels.

This figure represents direct AI/tooling spend over the snapshot window. It does not represent total company burn, founder time, opportunity cost, infrastructure spend,

review/recovery effort, or the discipline cost of building the substrate itself.

---

8. Reading Boundaries

The data above shows a specific shape. It does not show others.

What the data shows. A two-person team produced a production-shaped platform with substantial implementation, contract, infrastructure, and deployment surfaces in roughly six months. Coordination between human and AI participants sustained bounded autonomy with planning-side input at the scale of thousands of sessions and millions of tool-call records. Governance is encoded in CI gates, packet lifecycles, an invariant registry, and a 279,656-line compiled control-plane binary that humans, agents, and CI all invoke — not in social memory. The construction is live and operationally responsive at the snapshot date.

What the data does not show. External enterprise customer adoption. Compliance attestation (SOC2, ISO, third-party security audit). Disaster-recovery restore drill evidence. Formal SLO/error-budget/oncall maturity. Multi-team human organizational scaling. Uniform service maturity across all sixteen backend services. Whether the same substrate would sustain a fifty-person organization unchanged.

The next snapshot will answer different questions than this one.

— Christian

Appendix A. Reproduction

All measurements above are reproducible. Run from the repository root unless otherwise noted.

A.1 Repository state

```
git rev-parse HEAD
git log --all --until='2026-05-06T23:59:59-07:00' --oneline | wc -l
git log --all --format='%aI' | sort | sed -n '1p;$p'
git log --all --format='%an' | sort | uniq -c | sort -rn
```

A.2 GitHub PRs and workflow runs

```
gh api -X GET search/issues \
  -f q='repo:Eigenly-Inc/fireweed-platform is:pr is:merged merged:<=2026-05-06' \
  --jq .total_count
```

```
gh api -X GET /repos/Eigenly-Inc/fireweed-platform/actions/runs \
  -f created='2026-04-01..2026-04-30' -f per_page=1 -q '.total_count'
```

A.3 Implementation lines

```
find . -name '*.rs' -not -path './.git/*' -not -path './bazel-*/*' \
  -not -path '*/target/*' -print0 \
  | xargs -0 wc -l | awk '$2!="total"{s+=$1} END{print s}'

find . -name '*.ts' -not -path './.git/*' -not -path './bazel-*/*' \
  -not -path '*/node_modules/*' -not -path '*/target/*' -print0 \
  | xargs -0 wc -l | awk '$2!="total"{s+=$1} END{print s}'
```

A.4 System of Work corpus

```
find .work -name '*.md' -type f -print0 \
  | xargs -0 wc -l | awk '$2!="total"{s+=$1} END{print s}'

find .work -type f -print0 \
  | xargs -0 wc -l | awk '$2!="total"{s+=$1; c++} END{print c, s}'

find .work/system-of-work/domains -path '*/workflows/*.md' -type f | wc -l

jq '.invariants | length' .work/decision-records/invariant-registry.json
```

A.5 Mainline timeline

```
python3 local-reports/scripts/git_history_timeline_audit.py \
  --main-ref origin/main \
  --timezone America/Los_Angeles \
  --out local-reports/git-history-timeline-metrics-2026-05-07.json
```

A.6 Local Codex telemetry

Snapshot SHA-256: `c0babfb4373f108c15548fe7c6e314933dc34d7c442a53ec376668ab692ac391` at `.local/codex_audit/state_5.sqlite.snapshot.2026-05-07T09-08-26Z`. Generated artifacts: `local-reports/codex-thread-metrics-2026-05-07.json`, `local-reports/codex-thread-runtime-autonomy-metrics-2026-05-07.json`, `local-reports/codex-sqlite-turn-runtime-audit-2026-05-07.json`. Reproduction:

```
python3 local-reports/scripts/codex_sqlite_turn_runtime_audit.py \
  --snapshot .local/codex_audit/state_5.sqlite.snapshot.2026-05-07T09-08-26Z \
```

```
  --event-cutoff-utc 2026-05-07T09:08:26Z \
  --out local-reports/codex-sqlite-turn-runtime-audit-2026-05-07.json
```

A.7 Local Claude Code telemetry

Generated artifact: `local-reports/claude-code-thread-audit-2026-05-07.json`. Snapshot cutoff: `2026-05-07T21:01:06Z`. Reproduction:

```
python3 local-reports/scripts/claude_code_thread_audit.py \
  --snapshot-cutoff 2026-05-07T21:01:06Z \
  --out local-reports/claude-code-thread-audit-2026-05-07.json
```

A.8 Live public checks

```
for url in https://api.eigenly.com/health \
           https://assets.eigenly.com/health \
           https://eigenly.com \
           https://fireweed.eigenly.com \
           https://wishflower.eigenly.com; do
  printf '%s\t' "$url"
  curl -L -sS -o /tmp/body \
    -w '%{http_code}\t%{time_total}\t%{url_effective}\n' \
    --max-time 15 "$url"
done

for host in eigenly.com api.eigenly.com fireweed.eigenly.com wishflower.eigenly.c
  printf '%s\t' "$host"
  dig +short "$host" A | paste -sd ',' -
done
```

A.9 Governance/CI toolchain

Generated artifact: `local-reports/governance-toolchain-audit-2026-05-07.json`. Reproduction:

```
python3 local-reports/scripts/governance_toolchain_audit.py \
  --root . \
  --out local-reports/governance-toolchain-audit-2026-05-07.json
```

A.10 AI/tooling spend

Generated artifacts: `local-reports/ai-spend-ledger-2026-05-07.json` (transcribed transaction ledger) and `local-reports/ai-spend-summary-2026-05-07.json` (computed totals). Reproduction:

```
python3 local-reports/scripts/ai_spend_audit.py \
  --ledger local-reports/ai-spend-ledger-2026-05-07.json \
  --out local-reports/ai-spend-summary-2026-05-07.json
```

The full audit (`local-reports/fireweed-platform-unified-evidence-audit-2026-05-07.md`), with method disclosures, prior-draft corrections, and exhaustive per-section reproduction commands, is preserved separately as the backing document.