ACCELERATE **CHICAGO**

Craft software with confidence in 2026

Self-Building Software

Lessons from the HydraFlow Experiment

When no one reads the code where does the engineer go?

Travis Frisinger
Head of Agentic AI @ 8th Light

What we'll cover

1. The experiment: can a machine do real engineering, not just write code?
2. The machine structure
3. What makes the code safe
4. What broke and the gaps it opened

Most people talk about using AI to generate code faster.

This one isn't.

Can a machine produce software with real engineering rigor?

Not just generate code. Plan it, test it, review it, govern it.

Before HydraFlow, there was the knowledge problem.

We were already building around context, retrieval, project memory, delivery knowledge. That work became the soil HydraFlow grew out of.

InsightMesh

A real platform, built end to end by AI: *agents, retrieval, scheduled tasks, observability.*

Three years of AI-native building leading to it

I built HydraFlow.

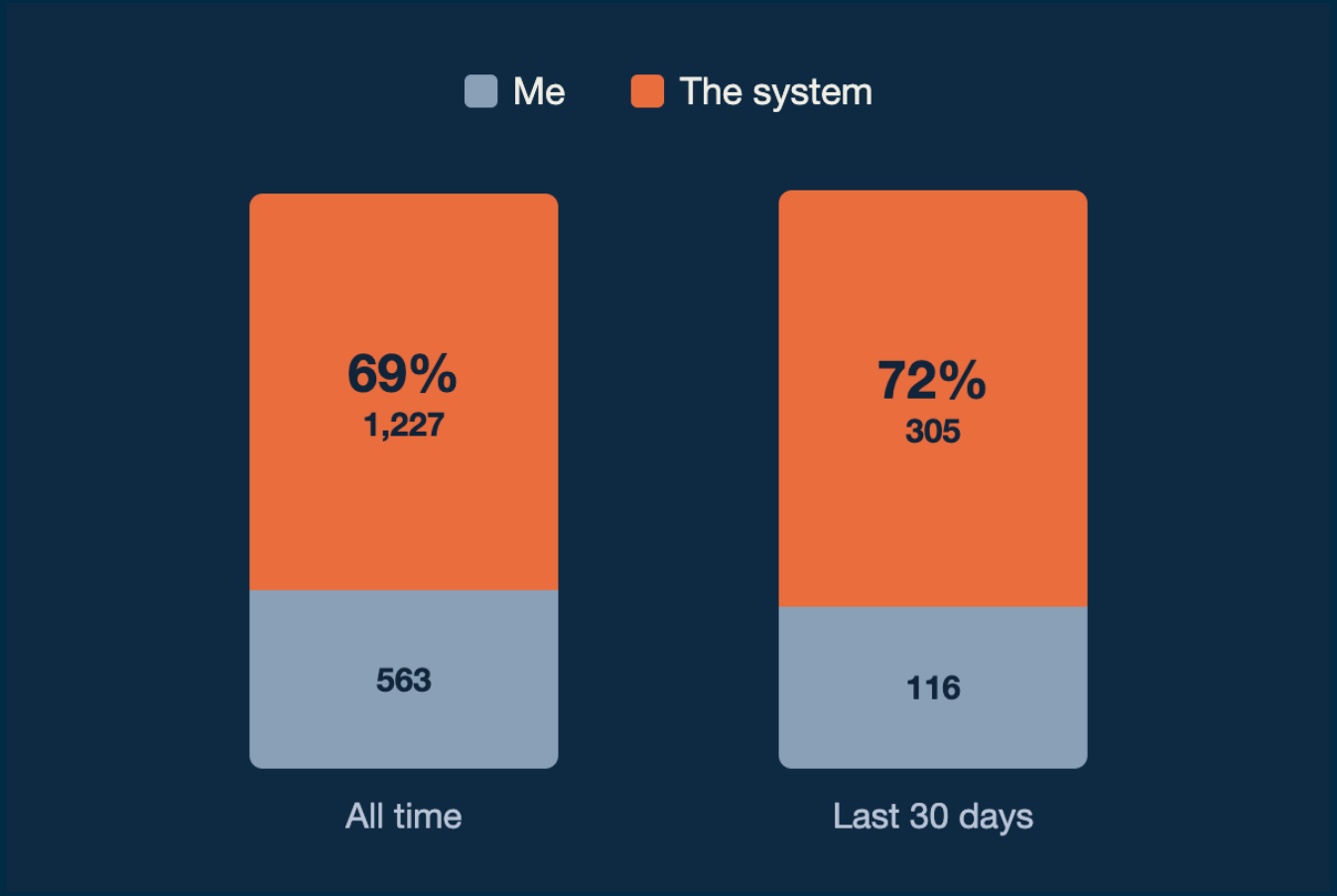
Intent in. Software out.

47 loops.
1,700+ merged PRs.

Building itself.

No human reviews its code. None ever has.

Two of every three commits are the system's.



I don't trust AI because it writes code.

I trust it because it runs a delivery system:

intent capture and expansion, context control, bounded execution, review, observability, rollback

TAKE THIS HOME

The craft underneath

6 **Observability**
the system shows its own work: traces, cost, structured logs

5 **CI / DevOps**
one pipeline every repo inherits: make targets, gates, branch protection

4 **RPI — research → plan → implement → review**
the loop's work cycle, not one-shot prompting

↻ the work cycle

3 **TDD / BDD**
tests first; Given-When-Then points at a world, not syntax

2 **Domain-driven design**
one ubiquitous language, shared by code and docs

1 **Hexagonal architecture**
every edge is a port with a fake: testable, swappable

FOUNDATION

The machine is loops, labels, and gates.

Issues enter through labels.

Loops claim work.

Gates decide what can move.

Dashboards expose state.

Code was identity.

I know what that sentence does to a room like this.

I taught TDD to hundreds of engineers. Red, green, refactor was how I proved the work was good.

The keyboard wasn't just a tool. It was the identity.

The bet had receipts.

- **FLUID, Mar 2025: code designed to mutate**
- **AI Decision Loop, Mar 2025: 2,500 conversations analyzed**
- **Coherence, Apr 2025: plausibility isn't truth**

The craft had to be encoded first.

- **F** Flexible Composition: loosely coupled, swappable parts.
- **L** Live Prototypes: build live, then harden with tests and CI.
- **U** Unified Context: reduce ambiguity, feed the system meaning.
- **I** Intent-Driven Structure: names and comments that carry the why.
- **D** Dynamic Refactorability: expect change, make it easy to reshape.

How much of the human can you pull out and what breaks when you do?

This was the experiment.

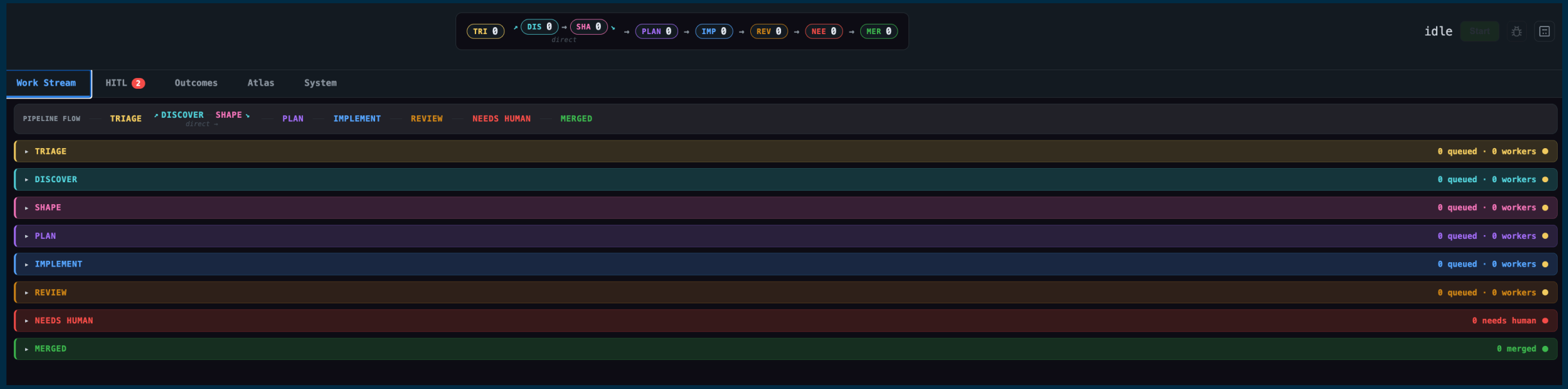
Steve Yegge was asking the same thing out loud. He called his Gas Town.

It started as a folder.

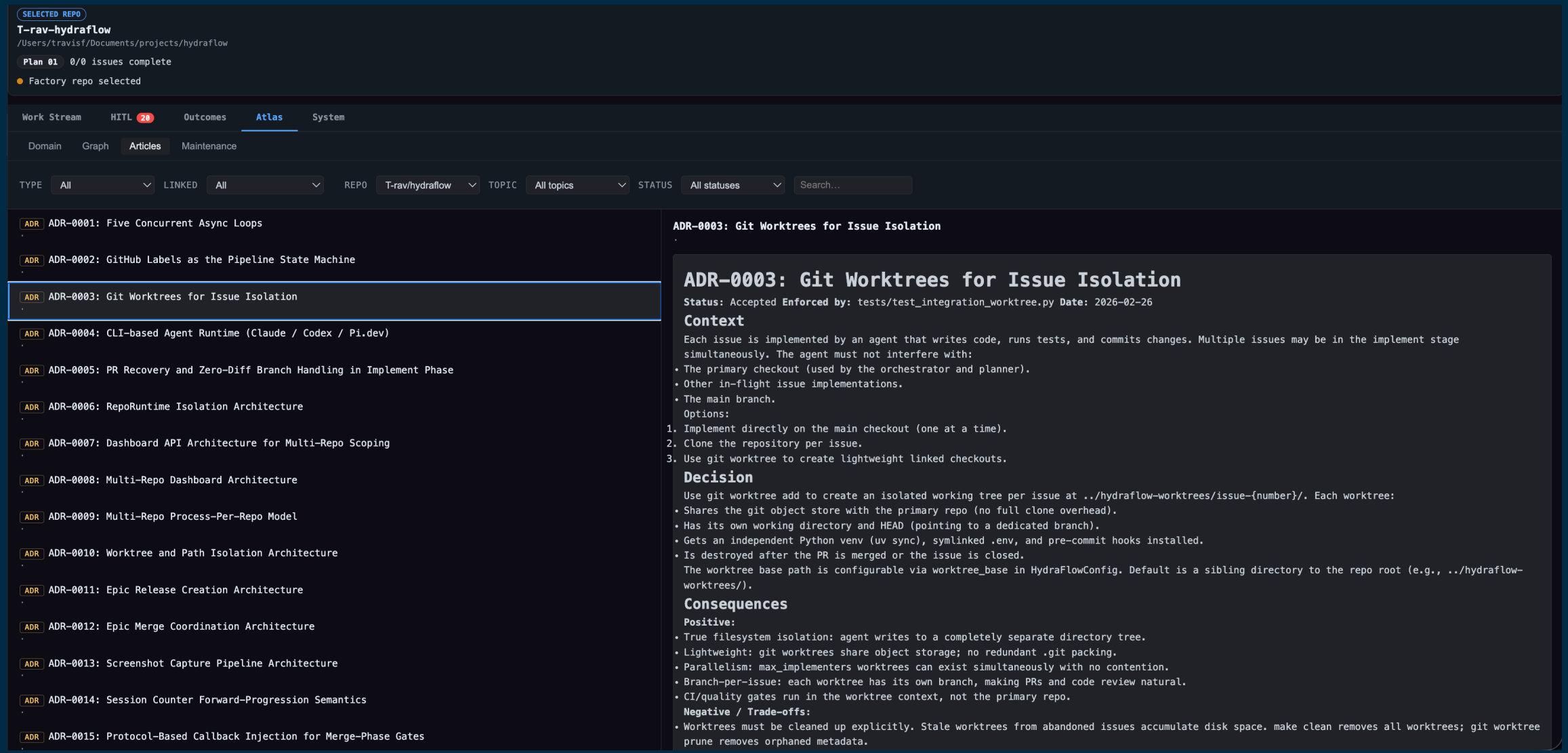
dx/hydra, a folder inside the platform.

Three days later it left, into its own repo, and it's been building itself ever since.

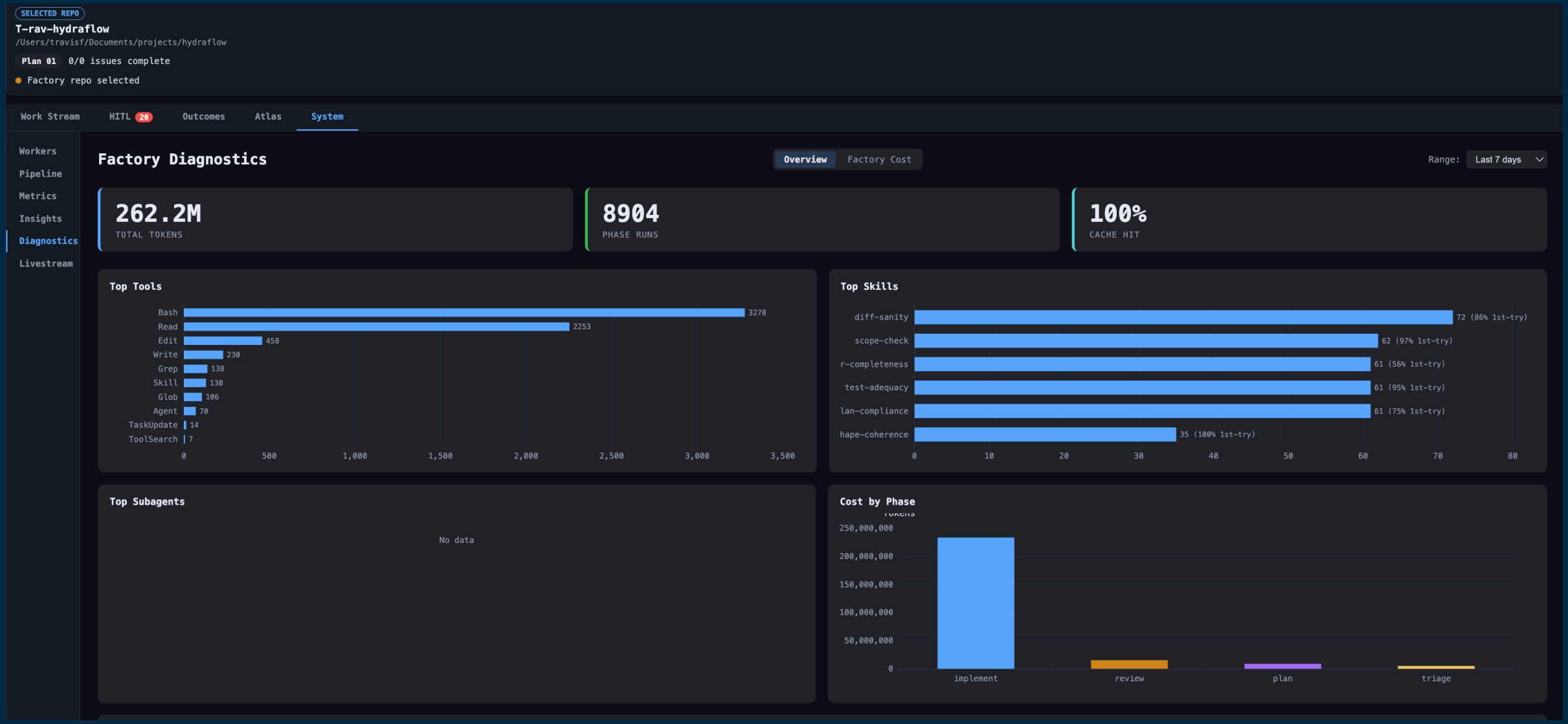
Work stream


TRI 0
DIS 0
SHA 0
direct
PLAN 0
IMP 0
REV 0
NEE 0
MER 0
idle
Start
Work Stream
HITL 2
Outcomes
Atlas
System
PIPELINE FLOW
TRIAGE
DISCOVER
SHAPE
direct
PLAN
IMPLEMENT
REVIEW
NEEDS HUMAN
MERGED
TRIAGE 0 queued · 0 workers
DISCOVER 0 queued · 0 workers
SHAPE 0 queued · 0 workers
PLAN 0 queued · 0 workers
IMPLEMENT 0 queued · 0 workers
REVIEW 0 queued · 0 workers
NEEDS HUMAN 0 needs human
MERGED 0 merged

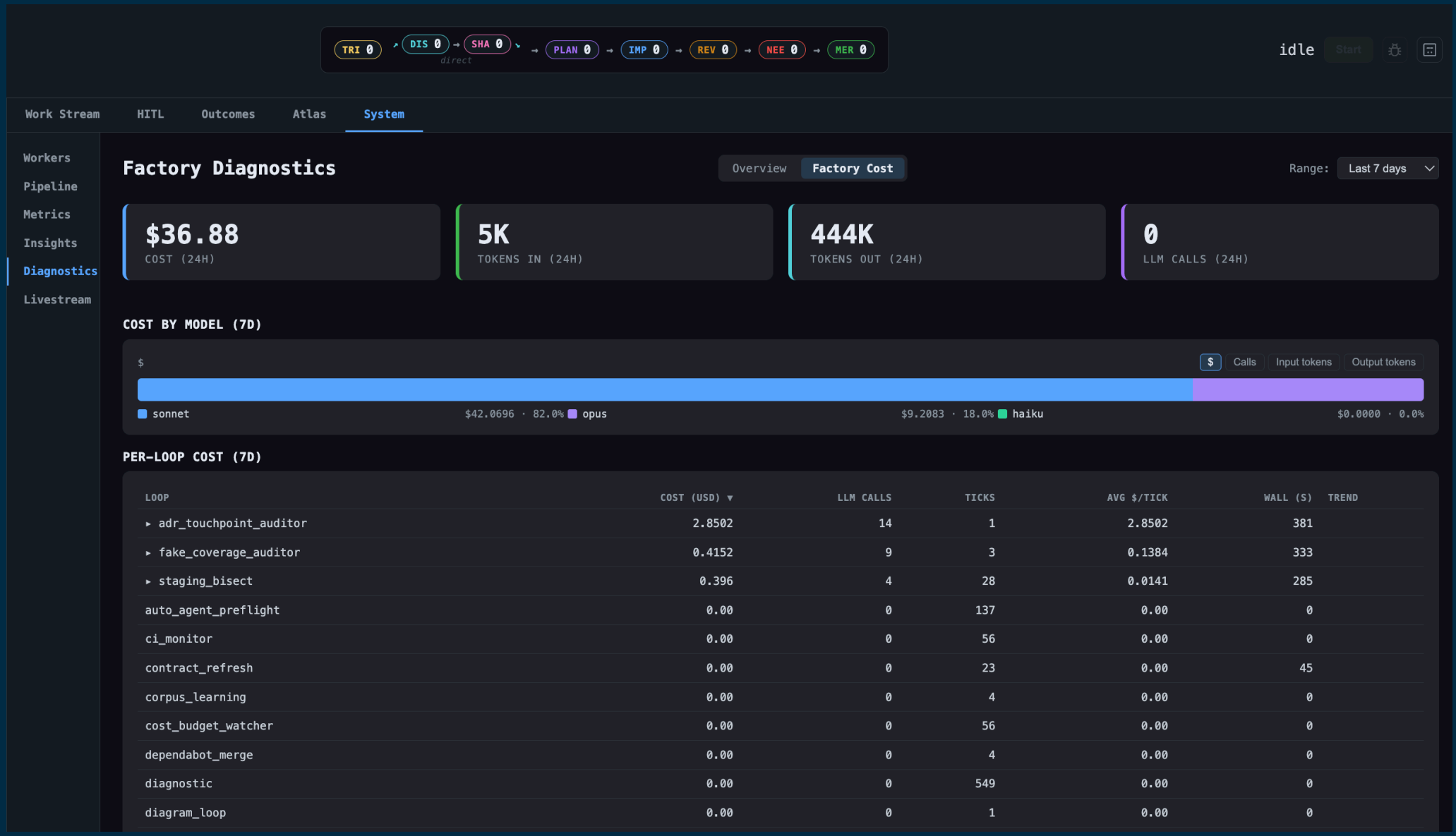
ADRS


SELECTED REPO
T-rav-hydraflow
/Users/travisf/Documents/projects/hydraflow
Plan 01 0/0 issues complete
Factory repo selected
Work Stream
HITL 20
Outcomes
Atlas
System
Domain
Graph
Articles
Maintenance
TYPE All
LINKED All
REPO T-rav/hydraflow
TOPIC All topics
STATUS All statuses
Search...
ADR ADR-0001: Five Concurrent Async Loops
ADR ADR-0002: GitHub Labels as the Pipeline State Machine
ADR ADR-0003: Git Worktrees for Issue Isolation
ADR ADR-0004: CLI-based Agent Runtime (Claude / Codex / Pi.dev)
ADR ADR-0005: PR Recovery and Zero-Diff Branch Handling in Implement Phase
ADR ADR-0006: RepoRuntime Isolation Architecture
ADR ADR-0007: Dashboard API Architecture for Multi-Repo Scoping
ADR ADR-0008: Multi-Repo Dashboard Architecture
ADR ADR-0009: Multi-Repo Process-Per-Repo Model
ADR ADR-0010: Worktree and Path Isolation Architecture
ADR ADR-0011: Epic Release Creation Architecture
ADR ADR-0012: Epic Merge Coordination Architecture
ADR ADR-0013: Screenshot Capture Pipeline Architecture
ADR ADR-0014: Session Counter Forward-Progression Semantics
ADR ADR-0015: Protocol-Based Callback Injection for Merge-Phase Gates
ADR-0003: Git Worktrees for Issue Isolation
ADR-0003: Git Worktrees for Issue Isolation
Status: Accepted Enforced by: tests/test_integration_worktree.py Date: 2026-02-26
Context
Each issue is implemented by an agent that writes code, runs tests, and commits changes. Multiple issues may be in the implement stage simultaneously. The agent must not interfere with:
• The primary checkout (used by the orchestrator and planner).
• Other in-flight issue implementations.
• The main branch.
Options:
1. Implement directly on the main checkout (one at a time).
2. Clone the repository per issue.
3. Use git worktree to create lightweight linked checkouts.
Decision
Use git worktree add to create an isolated working tree per issue at ../hydraflow-worktrees/issue-{number}/. Each worktree:
• Shares the git object store with the primary repo (no full clone overhead).
• Has its own working directory and HEAD (pointing to a dedicated branch).
• Gets an independent Python venv (uv sync), symlinked .env, and pre-commit hooks installed.
• Is destroyed after the PR is merged or the issue is closed.
The worktree base path is configurable via worktree_base in HydraFlowConfig. Default is a sibling directory to the repo root (e.g., ../hydraflow-worktrees/).
Consequences
Positive:
• True filesystem isolation: agent writes to a completely separate directory tree.
• Lightweight: git worktrees share object storage; no redundant .git packing.
• Parallelism: max_implementers worktrees can exist simultaneously with no contention.
• Branch-per-issue: each worktree has its own branch, making PRs and code review natural.
• CI/quality gates run in the worktree context, not the primary repo.
Negative / Trade-offs:
• Worktrees must be cleaned up explicitly. Stale worktrees from abandoned issues accumulate disk space. make clean removes all worktrees; git worktree prune removes orphaned metadata.

Tools

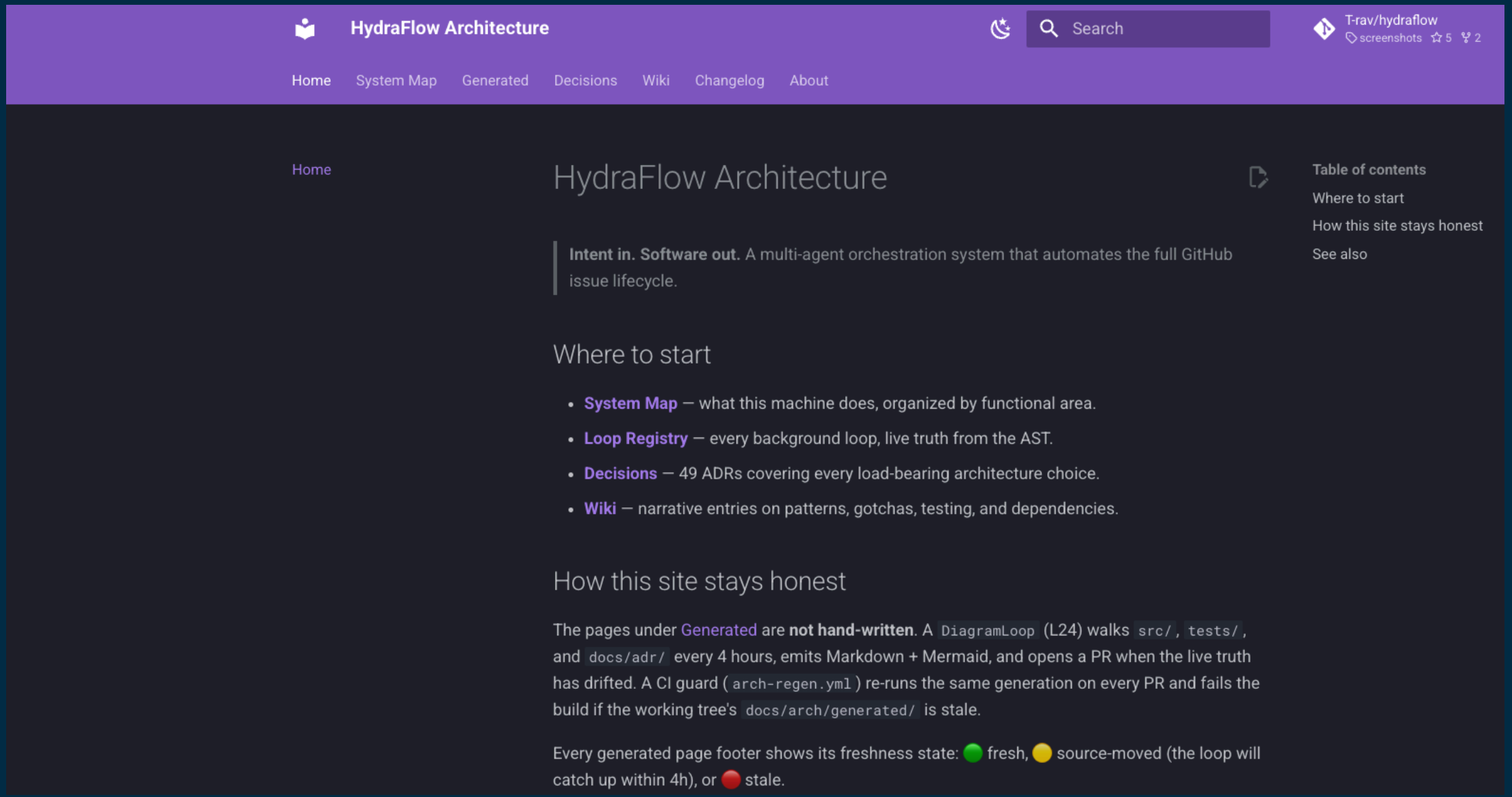

SELECTED REPO
T-rav-hydraflow
/Users/travisf/Documents/projects/hydraflow
Plan 01 0/0 issues complete
Factory repo selected
Work Stream
HITL 20
Outcomes
Atlas
System
Workers
Pipeline
Metrics
Insights
Diagnostics
Livestream
Factory Diagnostics
Overview
Factory Cost
Range: Last 7 days
262.2M
TOTAL TOKENS
8904
PHASE RUNS
100%
CACHE HIT
Top Tools
Bash 3278
Read 2253
Edit 458
Write 230
Grep 138
Skill 130
Glob 106
Agent 70
TaskUpdate 14
ToolSearch 7
0
500
1,000
1,500
2,000
2,500
3,000
3,500
Top Skills
diff-sanity 72 (86% 1st-try)
scope-check 62 (97% 1st-try)
r-completeness 61 (56% 1st-try)
test-adequacy 61 (95% 1st-try)
lan-compliance 61 (75% 1st-try)
hape-coherence 35 (100% 1st-try)
0
10
20
30
40
50
60
70
80
Top Subagents
No data
Cost by Phase
250,000,000
200,000,000
150,000,000
100,000,000
50,000,000
0
implement
review
plan
triage

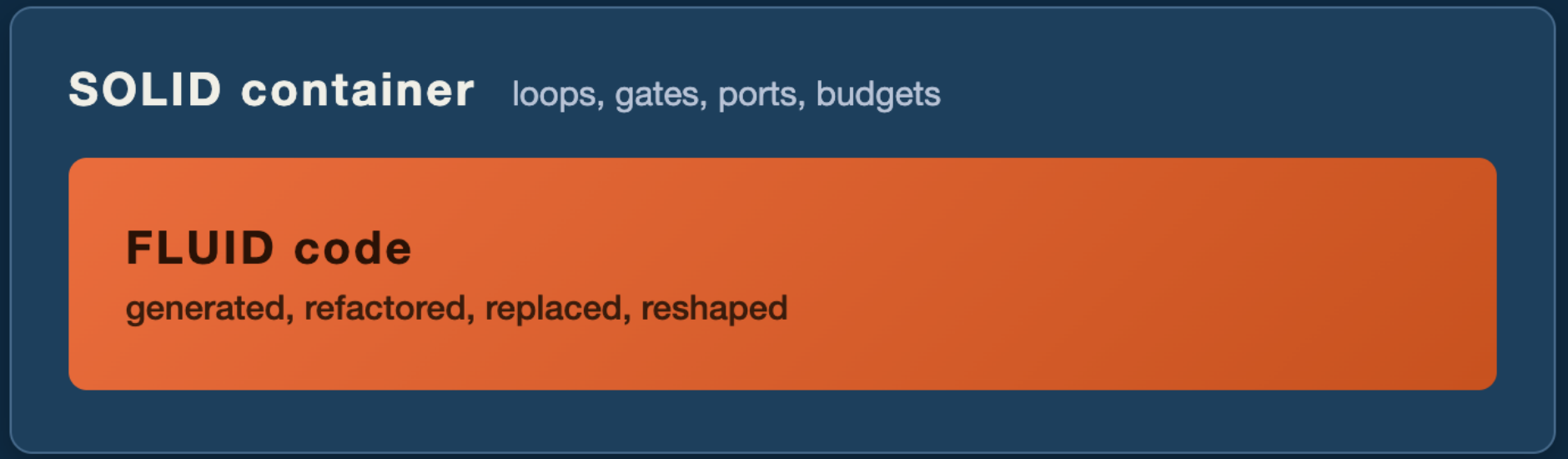
Cost



LOOP	COST (USD) ▼	LLM CALLS	TICKS	AVG $/TICK	WALL (S)	TREND
▸ adr_touchpoint_auditor	2.8502	14	1	2.8502	381	
▸ fake_coverage_auditor	0.4152	9	3	0.1384	333	
▸ staging_bisect	0.396	4	28	0.0141	285	
auto_agent_preflight	0.00	0	137	0.00	0	
ci_monitor	0.00	0	56	0.00	0	
contract_refresh	0.00	0	23	0.00	45	
corpus_learning	0.00	0	4	0.00	0	
cost_budget_watcher	0.00	0	56	0.00	0	
dependabot_merge	0.00	0	4	0.00	0	
diagnostic	0.00	0	549	0.00	0	
diagram_loop	0.00	0	1	0.00	0	

Wiki - https://t-rav.github.io/hydraflow/





The code can mutate because the boundaries don't.

“I don't prompt Claude anymore. I have loops that are running, they're the ones prompting Claude. My job is to write loops.”

Boris Cherny, Head of Claude Code June 2, 2026

The loop categories.

- Intake: turns signals, errors and requests, into tracked work
- Autonomy: does the work, and fixes its own failures
- Operations: keeps the pipeline flowing, unsticks what jams
- Release: promotes to staging and main, bisects bad cuts
- Repo Health: CI, security, flakes, cleanup, drift
- Learning and Insights: retrospectives, wiki, corpus learning
- Governance and Audit: branch protection, gate activation, audits
- Meta-Observability: watches the watchers

The category that matters most: it red-teams itself.

Builds adversarial cases from past failures.

They challenge the system's assumptions before reality does.

It doesn't just test for success. It manufactures doubt.

Lesson: Quality moved from social to structural.

Senior reviewers, tribal knowledge, manual discipline: none of it scales to a system that mutates daily.

The code got disposable. The guardrails got precious.

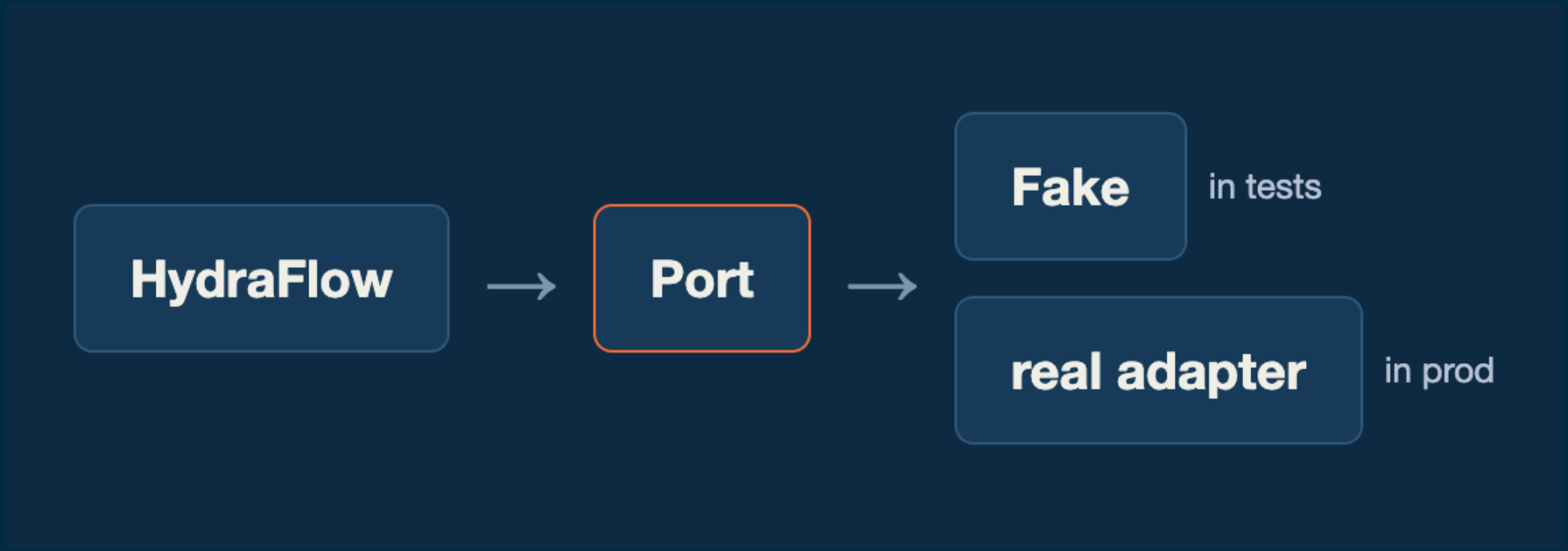
The encoding has a size

- **15,600+ tests**
- **70% coverage floor, CI-enforced**
- **478 world-scenario tests**
- **590+ regression tests from bugs that already bit us**
- **It can change the code. It can't change the rules that bound the damage.**

Testing this used to mean an environment.

A staging GitHub org. Real tokens. Rate limits. Flaky networks. An LLM bill.

MockWorld: the test chamber



Given-When-Then was always pointing at the world.

Given a world state.

When a loop acts.

Then the world changes in a specific way.

BDD was always about observable change, not pretty syntax.

It's now a delivery system, not a code generator. That's the worry.

A system that maintains itself opens gaps.

Judgment. **Comprehension.** **Erosion.**

Judgment gap: it stays skeptical of itself.

- Advisor pattern: a second opinion wired into review.
- Corpus-learning loop: past failures become fresh attacks.
- Provocateurs: weekly adversarial runs.
- Artifact: tests that resist the system's own future laziness.

Comprehension gap: I don't read the code, I read what it says about itself.

- **300+ wiki entries, regenerated from live pipeline events**
- **90 ADRs**
- **architecture diagrams regenerated on every PR**
- **This isn't documentation after the fact.**

If the code and the glossary disagree, the build fails.

Meaning gap: the vocabulary is load-bearing.

41 domain terms live as files in the repo; the glossary auto-extracts from code.

Erosion gap: a softened gate looks like a routine change.

A CI workflow named gates-drift checks that the hard limits still match their source of truth.

Next frontier: *an independent exam it can't tamper with, to catch it lowering its own standards.*

I'm not out here alone.

- Some trust reruns. Some trust tests.
- Some keep a gate. Some babysit the fleet.
- I trust the standard: craft, encoded and enforced.
- Different bets on one question: what holds when you step back?

For bounded, legible software: more than I expected.

The human doesn't disappear. It moves into product and system judgment.

Can the child maintain the parent?

The next experiment: HydraFlow goes home to maintain the platform that birthed it.

Lesson: When making is cheap, judgment is scarce.

The constraint stops being: can we build it?

It becomes: should we build it, what does it solve and who will look after it?

Encode the craft.
Guide the machine.

Your craft, encoded as guardrails on top of established patterns and practices.

Hand over the keystrokes. Keep the judgment.

https://hydraflow.ai · https://aibuddy.software · https://tddbuddy.com · https://linkedin.com/in/Travis-frisinger

Questions?