NFC Summit Lisbon • June 4, 2026

AI-Assisted Software Engineering

Building Moltis as a solo founder

Fabien Penso · https://pen.so · @fabienpenso

https://www.moltis.org

Who I am

I have been building internet products, infrastructure, and developer tools since the late 1990s.

FOUNDER / CTO

LinuxFr, Stuart, Kard

Open-source communities, logistics, and fintech products used at scale.

PRINCIPAL ENGINEER

Beam, Constellations, Dango

Rust systems, encrypted sync, Cosmos infrastructure, exchanges, and production indexing for Stargaze.

SOLO BUILDER

Constellations -> Moltis

Two serious solo Rust projects: before and after daily AI-assisted engineering.

This talk comes from day-to-day AI-assisted work on software I have to maintain and ship. Later this month I am joining Microsoft AI and moving to California.

What Moltis is

A **local-first persistent personal agent server**: one Rust binary between you, your tools, your memory, your channels, and multiple LLM providers.

GITHUB STARS
2.7K
moltis-org/moltis

ISSUES
367
49 open · 318 closed

PULL REQUESTS
679
12 open · 667 closed

MERGED PRS
566
shipping velocity

GATEWAY
One place for agents
Streaming chat, provider routing, coding agents, tools, sessions, and APIs.

MEMORY
Persistent context
Durable sessions, long-term memory, workspace files, project context, hooks, and cron.

CONTROL
Runs on your machine
Password/passkey auth, encrypted vault, sandboxing, local data, and self-hosted deploys.

Before AI: Constellations was not a toy

A production Cosmos indexer I built before the current AI-assisted workflow.

WORKLOAD

3,500h

about 2 years full-time

CODEBASE

118K

current tokei LoC

TRAFFIC

15M+

Stargaze requests/day

What it did

Indexed CosmWasm contracts and on-chain data for Stargaze, Osmosis, Neutron, Noble, dYdX, Kujira, and more.

Why it mattered

Stargaze pages went from loading in **>30 seconds** to **<500ms** through a real-time API.

This is my baseline for "before AI": a serious solo Rust infrastructure project, running real production traffic.

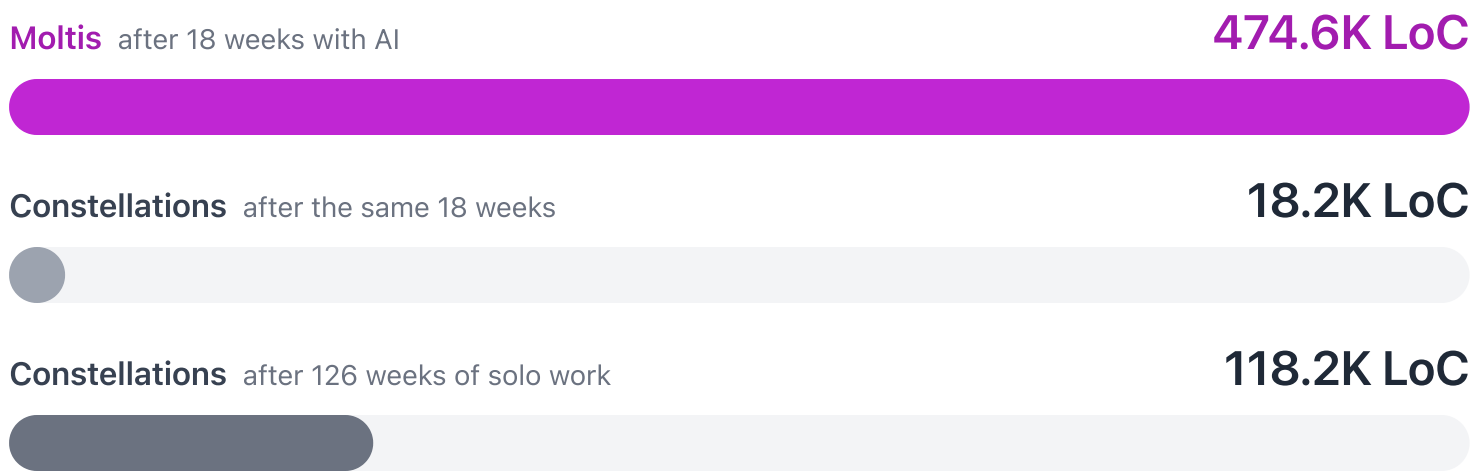
What changed in output

Same person. Same solo-founder pattern. Plain `tokei` code lines in checked-out repositories.

Codebase size by elapsed project time

LINEAR SCALE

Same 18-week window: **26.1x more LoC** with Moltis



Rewrites are no longer sacred

The cost of generating code is collapsing. The cost that remains is deciding, verifying, and maintaining it.

Bun: Zig → Rust

first → last commit: 9d 18h

BUN PR #30412

6,755

commits merged into main for the Zig → Rust rewrite.

BUN PR #30412

1M

lines rewritten across 2,188 changed files.

TEST SUITE

99.8%

of Bun's pre-existing tests reportedly passing.

Jarred Sumner wrote that the port fixed memory leaks and flaky tests, and gave Bun compiler-assisted tools for memory bugs that had cost enormous debugging time.

My actual workflow

HUMAN LOOP

1. Define the user outcome
2. Ask AI to inspect the codebase first
3. Make the smallest correct change
4. Review every diff
5. Run tests, linters, and builds
6. Ship, then listen to users

COLLABORATION

Share prompts, not patches

Other solo founders ask similar things. A prompt is faster to share than a PR, and safer because I can rerun it through my guardrails.

AI WORKFLOW

I barely touch the IDE

Most changes start from agent sessions now. I open the IDE only a few times a week.

My parallel issue workflow

The goal is not one giant AI session. It is many isolated, reviewable workstreams.

QUEUE

Issues first

Bugs first, because I want Moltis to be stable.

ISOLATION

One workspace per PR

Separate workspace for each issue or pull request.

AGENTS

OpenCode + Claude

Both 20x memberships, more than 10B tokens/month.

ADVERSARIAL REVIEW

Agents check agents

One model reviews another for fixes or simpler approaches.

EXTERNAL REVIEWER

Greptile on GitHub

Agents work through feedback until it reaches 5/5.

HUMAN OWNER

Fast final review

I review, merge, then E2E and release gates catch regressions.

This keeps multiple issues moving at once. I am building **Polyphony** to automate more of this: https://polyphony.to

Add determinism around the AI

AI is non-deterministic. Surround it with deterministic systems so bad output cannot land quietly.

OLD LESSON

Stop debating taste

Spaces vs tabs taught me: put RuboCop, formatters, and linters in CI, then move on.

NEW LESSON

Bound the agent

In Moltis, CI is the immune system that keeps AI output from drifting away from the project.

LANGUAGE CHOICE

Rust helps AI

Types, ownership, and memory safety turn whole classes of mistakes into compiler errors.

Moltis CI is the immune system

These gates turn taste, architecture, and release risk into executable constraints.

Shape

rustfmt, Biome, i18n parity, install docs sync, 1,500-line file limit.

Rust behavior

Clippy with `-D warnings`, nextest, coverage, contract tests.

Frontend

TypeScript check, Vite build, Playwright E2E, no JS-error assertions.

Integrations

Live provider tests, provider E2E scenarios, OpenAI/Ollama/sandbox E2E.

Release gates

Provider release checks, package builds, tag validation, changelog guard.

Local parity

`local-validate.sh` publishes PR statuses: fmt, lint, test, E2E, macOS, iOS.

Release safety: what users install

Faster code increases the blast radius of a bad release. The artifact needs proof attached to it.

SIGNATURES

Sigstore + my GPG key

Keyless CI signatures plus detached maintainer GPG signatures for release assets.

INTEGRITY

Checksums + verification

SHA256/SHA512 checksums; `verify-release.sh` pins my GPG fingerprint.

SBOM

CycloneDX + SPDX

Release SBOMs are generated, signed, checksumed, uploaded, and attested.

ATTESTATIONS

GitHub provenance

Artifacts, container images, and SBOMs link back to the workflow run.

Release safety: where it came from

Supply-chain safety is not only signatures. It is also protecting the factory and recording source provenance.

CI PROTECTION

Zizmor guards workflows

CI, release, docs, Homebrew, and benchmark workflows are scanned for attack paths.

RELEASE GATES

Tests before packages

Clippy, tests, E2E, and live provider checks must pass before packaging.

SOURCE PROVENANCE

Pin GitHub origins

Skills/plugins pin GitHub commit SHAs; source drift requires re-trust.

AUDIT TRAIL

Leave evidence behind

Trust, install, source-drift, enable/disable, and dependency events are logged.

How I talk to the AI

Weak

```
Build auth
```

Useful

```
Add password reset using the existing auth middleware. Persist tokens in the current credential store. Add tests for expiry and invalid tokens. Keep existing routes compatible.
```

Prompting is delegation. Context is management.

The PR workflow is the next bottleneck

AI did not just make typing faster. It changed the volume, shape, and cadence of software work.

GITHUB OCTOVERSE 2025

65M -> 82M

monthly code pushes from 2024 to 2025, a **25% jump**.

GITHUB OCTOVERSE 2025

39.5M -> 47.5M

monthly PRs created from 2024 to 2025, **+20.4% YoY**.

GITHUB WORKFLOW REPORT

11.5B

Actions minutes running tests in 2025, **+35%** year over year.

Source: GitHub Octoverse 2025 and GitHub workflow report. Supplemental GitHub search snapshot: 2026 PR creation was 2.55x the same May week in 2025.

The old loop was designed for scarce code

Issue → branch → PR → human review → merge works when changes are relatively expensive.

NEW PROBLEM

attention scarcity

The tooling of the last 20 years needs to be rethought: review, ownership, validation, and product intent have to become more continuous than a GitHub PR page.

What changed for me as a solo founder

More surface area

I can touch frontend, backend, docs, tests, CI, and releases in one session.

Faster recovery

When I hit a compiler error or broken build, I lose less time getting unstuck.

Higher standards

Because code is cheaper, tests, docs, and polish are less optional.

The job becomes less "can I build this?" and more "is this worth building, and can I keep it coherent?"

Beginner rules that actually help

1. Learn enough code to review the output
2. Keep tasks small
3. Use version control from day one
4. Ask AI to explain the change
5. Run the app and test the unhappy paths
6. Never paste secrets casually
7. Prefer boring architecture
8. Let users, not demos, judge progress

If you cannot tell whether the result is good, slow down. The tool is not the adult in the room.

AI is great for builders, but it also exposes the next bottlenecks: good ideas, distribution, taste, and stamina. Coding was never the only hard part.

Thank you

Moltis

https://www.moltis.org

https://github.com/moltis-org/moltis

Fabien Penso · https://pen.so · @fabienpenso

Appendix: if the slot becomes 20 minutes

Show a real workflow

Issue, isolated workspace, agent pass, review feedback, tests, merge.

Open Moltis briefly

Use it only to make the talk concrete, not as a product tour.

Security guardrails

Explain why CI, signatures, SBOMs, and provenance matter more when code moves faster.

Audience Q&A

Ask what they are building and where AI currently breaks their workflow.