Elias Leslie — Detailed Portfolio

Last updated: 2026-06-09

Summary

Elias Leslie builds practical automation systems across AI automation, security automation, cyber security, agentic developer tooling, and full-stack infrastructure. The common thread is operator-grade software: documented setup, secure defaults, tests, runtime smoke checks, secret hygiene, clean install verification, and honest limitations.

The projects below are public repositories, presented by theme rather than ranked. Each is a standalone public release.

SummitFlow

Task orchestration and evidence capture for AI-assisted software development. Repository: https://github.com/elias-leslie/summitflow

- **Problem:** AI-assisted development scatters task state, quality gates, and verification evidence across one-off scripts and chat logs, so it is hard to see what was actually built, checked, and proven across projects.
- **Solution:** A local-first control plane that tracks tasks, subtasks, steps, and dependencies; runs quality gates and code-health scans; drives autonomous execution hooks and browser checks; and keeps operator-visible verification evidence. Built for developers running their own agent tooling, not as a hosted SaaS.
- **Stack:** FastAPI, Python 3.13, SQLAlchemy, Alembic, Next.js 16, React, TypeScript, PostgreSQL, Redis, Hatchet, pnpm, uv.
- **Built:** A FastAPI backend and Next.js operator UI for project/task state, Hatchet-driven scheduled and event-driven workflows, the st CLI for task/project/check/database/backup/browser/workflow operations, UI/API smoke-test evidence capture, and an Apache-2.0 public release with CI, a Docker Compose source stack, and a security policy.
- **Skills demonstrated:** Full-stack system design, workflow orchestration, CLI and developer-tooling design, runtime smoke verification, and public release discipline.
- **Security and AI relevance:** Keeps agent work local-first and auditable, and degrades clearly when optional integrations are absent instead of exposing credentials or crashing unrelated pages.
- **Status:** Public Apache-2.0 release for developers running their own agent tooling. Pairs with Agent Hub for routed AI completions and shared memory, but runs standalone.
- **Visual proof:** docs/images/summitflow-projects-dashboard.png is an inspected local projects-dashboard screenshot with no personal, customer, private infrastructure, token, or financial data.

Agent Hub

Self-hosted control plane for running, observing, and improving multi-provider AI agents. Repository: https://github.com/elias-leslie/agent-hub

- **Problem:** Most agent demos stop at chat. Running agents as real infrastructure needs provider routing, persistent memory, sessions, access control, and cost/latency visibility in one governed place.

- **Solution:** A self-hosted control plane that adds the operational layer: unified completions and streaming across configured providers, PostgreSQL-backed memory and context injection, named agents/personas, session history, request logs, routing telemetry, and operator dashboards.
- **Stack:** FastAPI, Python 3.13, Next.js 16, React, TypeScript, PostgreSQL with pgvector, Redis, Hatchet, pnpm, uv, plus an async Python SDK.
- **Built:** Provider adapters for Gemini, OpenAI, OpenRouter, Kimi, MiniMax, DeepSeek, xAI, and local OpenAI-compatible endpoints; memory/session/telemetry surfaces; client registration and access-control for companion apps; a Python SDK for completions, SSE streaming, and stateful sessions; and an Apache-2.0 release with public-safe screenshots, CI, and a security policy.
- **Skills demonstrated:** Multi-provider routing, credential boundaries, memory/session infrastructure, SDK design, operability and observability, and public release discipline.
- **Security and AI relevance:** Isolates provider credentials behind access-control surfaces, boots without provider keys, and fails clearly when optional integrations are unconfigured.
- **Status:** Public Apache-2.0 release. Works standalone and as SummitFlow's routed-agent backend. Expose beyond loopback only behind a reverse proxy with strong client/internal secrets.
- **Visual proof:** docs/images/agent-hub-dashboard.png is a public-safe command-center dashboard screenshot using demo data, inspected before release, with no tokens, personal, or financial data.

Security Hardening Automation (SHA)

Clean-room security hardening automation platform for Windows and Linux endpoints. Repository: https://github.com/elias-leslie/sha

- **Problem:** Endpoint hardening often becomes a brittle mix of one-off scripts, undocumented baseline assumptions, risky remote access, and unclear rollback paths.
- **Solution:** SHA models hardening as a bounded control plane: enroll endpoints, collect posture, browse curated controls, generate installer profiles, and route disruptive work through approval requests/grants.
- **Stack:** FastAPI, Python 3.13, SQLAlchemy, Pydantic, Next.js 16, React 19, TypeScript, Vitest, pnpm, uv, SQLite for local development.
- **Built:** Backend APIs, dashboard pages for fleet/endpoints/controls/installers/approvals, deterministic Linux/Windows bootstrap artifacts, generated JSON Schemas, Apache-2.0 public release docs, CI, and a clean public control-pack path.
- **Skills demonstrated:** Security automation design, public-source provenance cleanup, secret/history scanning, dependency vulnerability remediation, full-stack testing, browser/runtime smoke checks, and clean Proxmox install verification.
- **Security and AI relevance:** Keeps endpoint work bounded to typed hardening workflows, avoids arbitrary shell access, documents approval boundaries, and provides a foundation for supervised operator automation.
- **Status:** Early public control-plane/dashboard slice. It is not production-ready until authentication, authorization, production migrations/deployment hardening, and a completed privileged endpoint agent are added.
- **Visual proof:** docs/images/sha-control-plane-smoke.png is an inspected dashboard smoke-test screenshot with no personal, customer, private infrastructure, token, or financial data.

Aico

Linux desktop companion for terminal AI agents. Repository: https://github.com/elias-leslie/aico

- **Problem:** Terminal AI agents are useful but fragmented across shells, browser context, desktop selection, and project state.
- **Solution:** A Linux desktop companion with floating widgets, isolated tmux sessions, a local FastAPI sidecar, click-to-context capture, and optional browser/voice integrations.
- **Stack:** Electron, TypeScript, Vite, FastAPI, Python 3.13, tmux, uv, Node.js, browser extension APIs.
- **Built:** A desktop companion app, a local FastAPI sidecar, click-to-context capture, optional browser/voice integrations, a source installer, CI, and release hardening.
- **Skills demonstrated:** Desktop/Electron integration, local sidecar API design, tmux/session orchestration, browser-context capture, and release hardening.
- **Security and AI relevance:** Keeps sensitive workflow state local by default, uses loopback APIs, and degrades when optional integrations are absent.
- **Status:** Public source release for single-user Linux desktops; Wayland/global shortcut support varies by desktop environment.

A-Term

Self-hosted browser workspace for AI coding agents, shells, files, prompts, and notes. Repository: https://github.com/elias-leslie/a-term

- **Problem:** Agentic coding needs shells, files, prompts, and notes in one browser-accessible environment, and naive web terminals lose their session the moment the tab closes.
- **Solution:** A self-hosted browser workspace that runs multiple persistent, tmux-backed terminal sessions (Claude Code, Codex, Gemini CLI, Hermes, OpenCode, Pi, and shells) side by side in a resizable pane grid, with a file browser, a notes/prompt library, voice input, and full mobile support.
- **Stack:** FastAPI, Python 3.13, SQLAlchemy, Alembic, PostgreSQL, Next.js 16, React 19, TypeScript, xterm.js (WebGL), tmux, Tailwind CSS 4, pnpm, uv.
- **Built:** WebSocket PTY terminals over tmux for crash-proof sessions; up to six resizable panes with detach-to-window; per-pane dual shell/agent mode with built-in tool presets and custom tools; a sandboxed file browser with validated uploads; a notes/prompt library with version history and Agent-Hub-backed prompt cleaning; browser-native voice input; an on-screen keyboard and PWA install for mobile; and three auth modes (loopback/password/proxy) with security headers, CSP, and rate limiting.
- **Skills demonstrated:** Real-time WebSocket/PTY streaming with backpressure, terminal/session orchestration, full-stack developer-experience tooling, mobile/PWA support, and secure-by-default remote access.
- **Security and AI relevance:** Ships loopback-only by default, isolates the file browser against path traversal, and centralizes agent working context locally instead of in a hosted service.
- **Status:** Public project. Runs standalone, or pairs with SummitFlow and Agent Hub for shared projects, prompt cleaning, and a model catalog.

Portfolio AI

Self-hosted, full-stack investment intelligence workspace. Repository: https://github.com/elias-leslie/portfolio-ai

- **Problem:** Financial research workflows need repeatable ingestion, analysis, and reporting without exposing private holdings.
- **Solution:** A self-hosted, full-stack investment intelligence workspace that tracks portfolios and tax lots, scores a watchlist from market data, news, technicals, and fundamentals, computes a macro deployment gate, and optionally routes AI analysis through an Agent Hub companion — all on data you host.
- **Stack:** FastAPI, Python 3.13, SQLAlchemy, Next.js 16, React 19, PostgreSQL, Redis, Hatchet, pandas, scikit-learn, pandas-ta; yfinance, CBOE, FRED, and SEC EDGAR data plus optional paid market-data APIs.
- **Built:** Portfolio/tax-lot/transaction tracking with cost basis, P&L, tax-loss harvesting (wash-sale checks), and IPS drift/rebalance; a multi-pillar watchlist scorer with plain-language narratives; a macro deployment gate (FULL_DEPLOY/REDUCED/DEFENSIVE) with walk-forward and Monte Carlo backtests; ~63 cron-scheduled Hatchet workflows; household money, document-intake, budgeting, and retirement (Monte Carlo) surfaces with encrypted Plaid/SnapTrade linking; an AI investment-committee and thesis pipeline routed entirely through Agent Hub (no hardcoded model IDs); and a read-only MCP server.
- **Skills demonstrated:** Multi-source data pipelines, quantitative/technical analysis, lightweight ML, workflow orchestration at scale, full-stack reporting UI, and privacy-aware public documentation.
- **Security and AI relevance:** Boots without optional keys (degrading rather than failing), encrypts source and broker credentials at rest, keeps all LLM access behind Agent Hub, and uses only public claims with no real balances, holdings, transactions, account IDs, brokerage names tied to real data, or live portfolio values.
- **Status:** Public project; users must configure their own data sources and secrets.

Portfolio

Public Markdown/PDF project showcase with safe visual proof and links to released repos. Repository: https://github.com/elias-leslie/portfolio

- **Problem:** Public work needs a credible, hiring-focused hub that is polished without exposing proprietary/customer/private claims.
- **Solution:** A Markdown-first GitHub portfolio with project case studies, safe visual proof, PDF sources/exports, and links to public repos.
- **Stack:** GitHub README/Pages-style Markdown, maintainable PDF source, release screenshots/demo visuals.
- **Built:** Project case studies, maintainable PDF sources with a render script, safe visual assets, and public-claim hygiene.
- **Skills demonstrated:** Technical writing, public-claim hygiene, documentation tooling, and release-proof curation.
- **Security and AI relevance:** Curates public-safe visual proof and avoids proprietary, customer, or private claims.
- **Status:** Public portfolio repository; updated as projects are released.

Capability areas

- **AI automation and agentic tooling:** local-first agent workflows, prompt/session infrastructure, terminal/browser context capture, multi-provider control planes.
- **Security automation:** endpoint hardening, incident containment concepts, detection workflows, evidence exports, rollout/rollback discipline.

- **Cyber security engineering:** secure defaults, secret scanning, local API boundaries, install-time dependency verification, public release hygiene.
- **Developer tooling:** tmux/session orchestration, desktop/browser integrations, clean installers, CI, test suites, smoke tests.
- **Full-stack infrastructure:** FastAPI, Next.js/Electron, PostgreSQL/Redis-style backends, operator dashboards, Linux service/runtime integration.

Contact

LinkedIn: https://linkedin.com/in/elias-leslie

GitHub: https://github.com/elias-leslie