CLike

AI Native Pipeline for Product Engineers

From intent to impact.
CLike keeps developers in flow, augments delivery with agentic workflows, and bakes in governance, eval-driven quality, and a safe paved road for enterprises.

CLike - What is CLike

CLike is an **AI-native platform** that merges the **Harper-style pipeline** (I*DEA → SPEC → PLAN → KIT → FINALIZE*) with the **Vibe Coding philosophy** (intent/outcome-focused, developer in flow), and operationalizes it **with agentic workflows**, retrieval-grounded intelligence, **and human eval-driven quality gates** as **Gartner** suggestions.

Why it matters

- Flow state by default — minimize context switches; everything lives inside VS Code.
- Agentic & self-healing — AI assistants perform actions and auto-remediate (diffs, patches, tests).
- Enterprise paved road — governance, **auditability**, and **reproducibility** are built-in, not bolted on.

Inspired by the project's official Manifest and aligned with AI-Native SWE best practices.

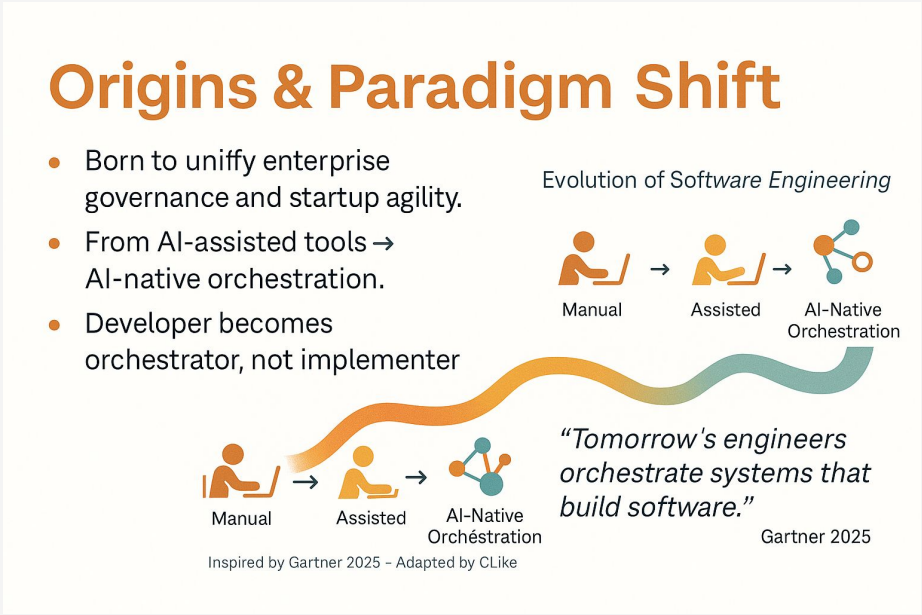
Clike - Where the Idea Comes From

- Harper (blog "Codegen Hero's Journey") → talks about a narrative workflow in which the LLM is a co-hero: starting with an idea, it generates a **SPEC** specification file, a **PLAN**, then a **KIT**, and iterating with short **feedback cycles**. This is the methodological and operational backbone of the solution definition phase. - Haprer

- Vibe Coding (Karpathy + Gartner) → emphasizes flow, intent, rapid prototyping, and cognitive offloading: the **developer becomes** a "**composer**" who works at the outcome level, not the code level. This has been incorporated by leaving the developer only with the design/intent steps and automating the build, testing, and security. Gartner Vibe

- AI-Native Software Engineering (Gartner) → introduces ***agentic workflows, autonomous improvement loops, human-in-the-loop, and security as a guardrail.*** The process includes SAST, DAST, UAT/E2E, make targets for automatic cycles → process exactly in line with these recommendations. Gartner AI

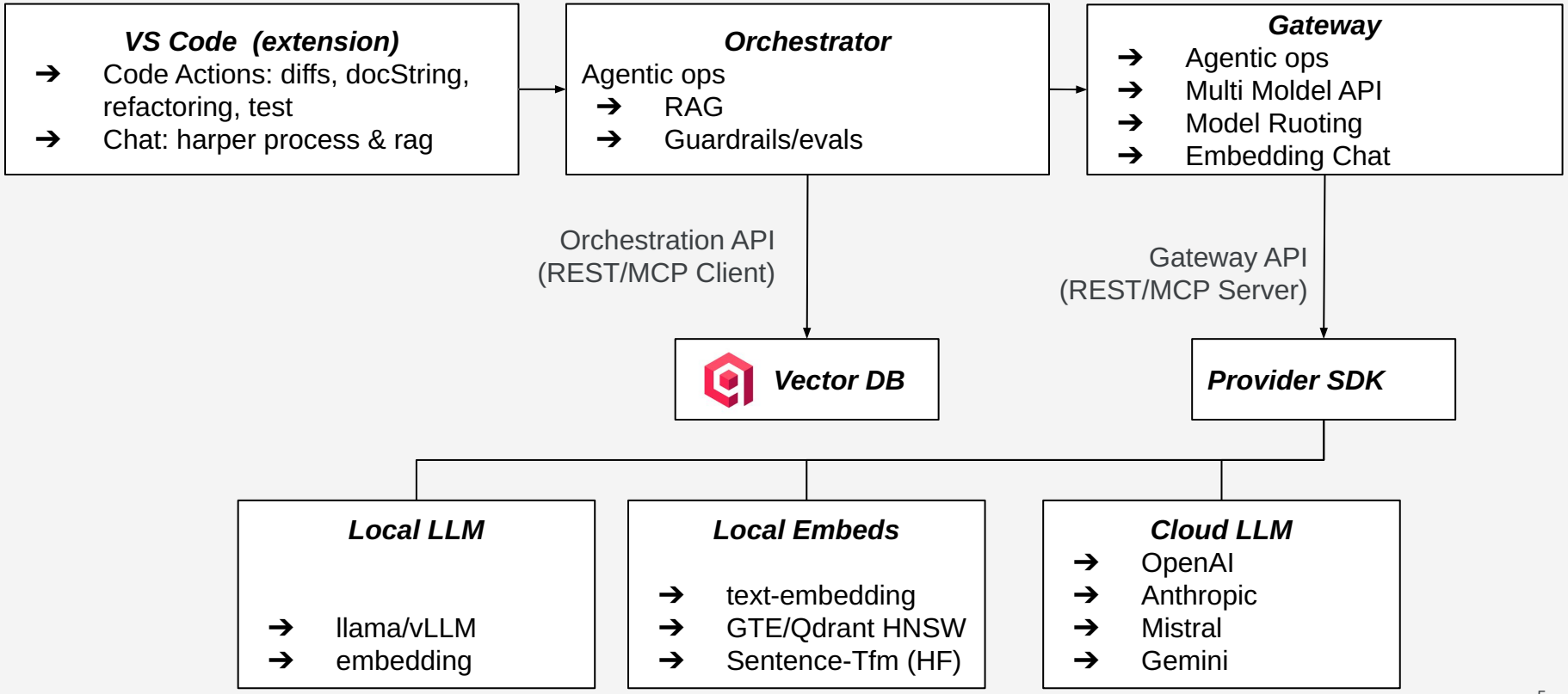
CLike - The Paradigm Shift



The new AI Paradgim

- From manual programming to the orchestration of AI agents and pipelines.
- Human-in-the-loop as a continuous validator(orchestrator.
- Focus on governance, explainability, and eval-driven gates.

CLike - Architecture



CLike - Harper Process an AI Native Pipeline

Harper transforms a single IDEA written by the developer into releasable software through short, testable, human-in-the-loop-governed phases.

- ➔ **IDEA** — Business & tech intent in one page; scope, value, constraints.
- ➔ **SPEC** — Measurable requirements and clear Definition of Done.Translate the IDEA into verifiable requirements and acceptance criteria that a machine and a human can both check.
- ➔ **PLAN** — Dependency-aware tasks with inputs/outputs and milestones. Output: `plan.json` + `PLAN.md` — clear inputs/outputs per task; ready for automated execution..
- ➔ **KIT** — Modular code + tests/docs aligned to the plan.
- ➔ **EVAL** — Automated checks: coverage, tests, lint/type, basics of security.
- ➔ **GATE** — Policy-as-code decision; human sign-offs and audit trail.
- ➔ **FINALIZE** — Version, changelog, release notes; learnings fed back to SPEC/IDEA.

Andrea Franco

November 5, 2025

CLike - Eval-Driven Governance

CLike makes tests adaptive: models derive what to test from **SPEC** and the plan, so quality stays current—not stale. The Eval Driven Governance si composed by three phases: A, Phase B an Phase C.

Phase A — PLAN → *ingredients*

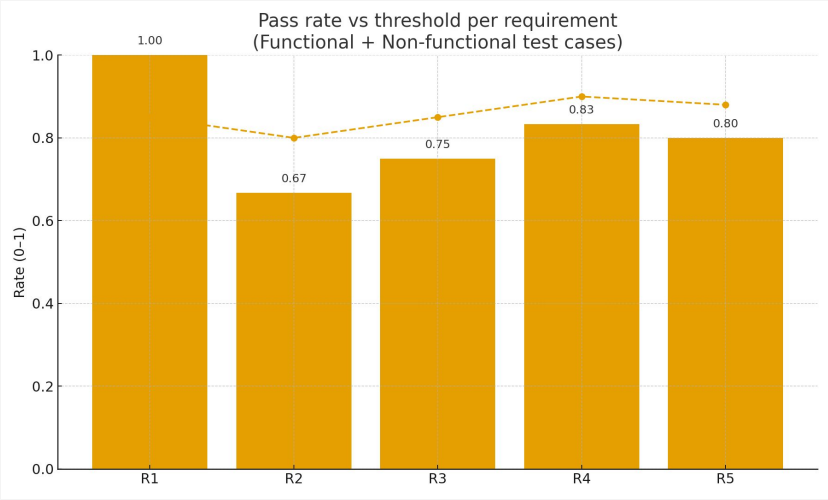
- Generate an **Ingredients Manifest** per lane: tools/adapters, datasets/fixtures, KPIs & thresholds, human checkpoints, env needs.

Phase B — KIT → *recipe*

- Assemble a **Validation Recipe** from the ingredients: concrete commands, test suites, data wiring, CI hooks, run order.

Phase C — EVAL & GATE → *cooking*

- **EVAL** “cooks” the recipe: runs adaptive tests (accuracy, explainability, trust, coverage, lint/type, perf, basic security).
- **GATE** enforces policy-as-code, collects HITL sign-offs, decides **promote/stop** with audit trail.



CLike - Roadmap

✅ COMPLETED

- ➔ VS Code Extension: Code Actions (Refactor, Docstring, Tests, Fix errors); Chat (Free / Harper / Coding); single-model & cross-model sessions; attachments + RAG (index & search).
- ➔ Harper Orchestrator: integrated RAG; commands /spec /kit /eval /gate /finalize (first working draft).
- ➔ LLM Gateway: Ollama (e.g., DeepSeek V2 Code), OpenAI, Anthropic; multi-model & model routing by profile; telemetry for token usage.
- ➔ Monitoring via Telemetry: usage={'input_tokens': 22172, 'cache_creation_input_tokens': 0, 'cache_read_input_tokens': 0, 'cache_creation': {'ephemeral_5m_input_tokens': 0, 'ephemeral_1h_input_tokens': 0}, 'output_tokens': 8192, 'service_tier': 'standard'} (→ input for **PVE**?)

🟡 IN PROGRESS

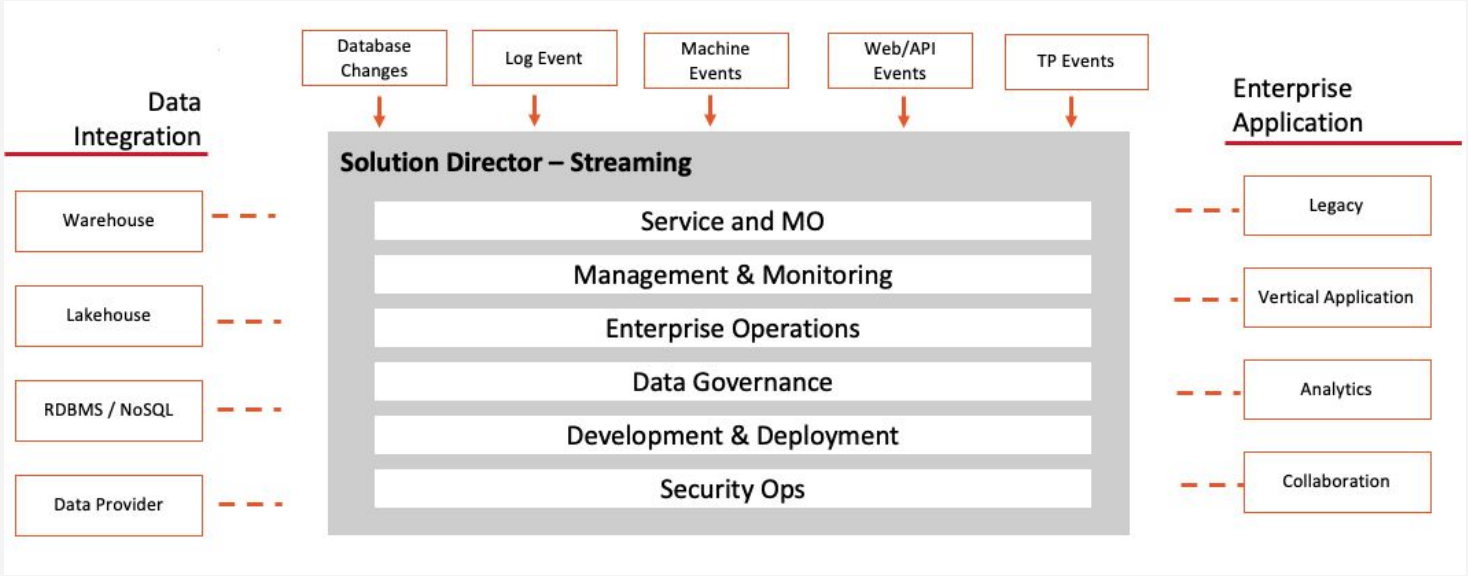
- ➔ Git end-to-end: version every phase (IDEA → FINALIZE) with PR templates and artifact traceability.

🔵 NEXT STEPS (short term)

- ➔ MCP (Model Context Protocol): standardize access to FS/Git, RAG, ...; make tools & context portable across environment.
- ➔ External Agents (optional but strategic): expose Harper steps as services/agents with clean I/O fo rnay phases to enable:
  - ◆ interoperability (other tools/teams can invoke steps
  - ◆ scalability (parallel runs, multi-model routing)
  - ◆ governance (tracing, retries, policy-as-code outside the IDE)

CLike - Domain Architecture Meets AI-Native Orchestration

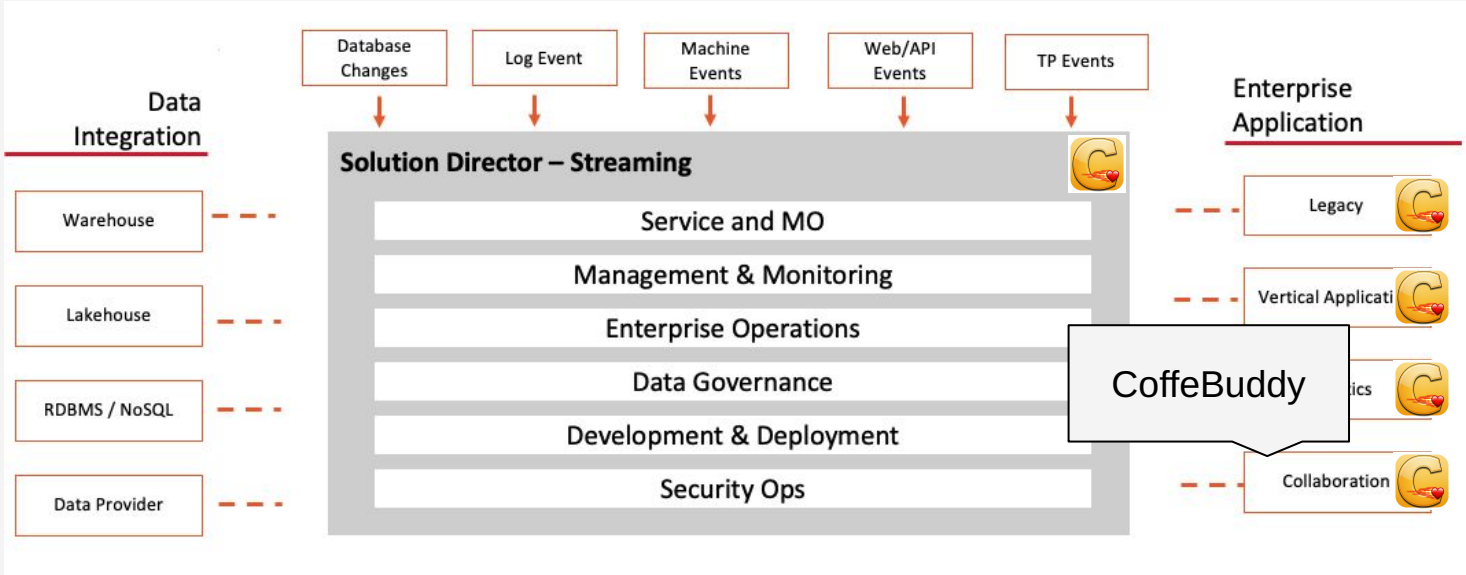
In **domain-driven architectures**, complexity is managed by *dividing problems into autonomous, governed domains.*



CLike - Domain Architecture Meets AI-Native Orchestration

In **domain-driven architectures**, complexity is managed by *dividing problems into autonomous, governed domains.*

Clike extends this principle to AI-native software engineering: each ***domain*** *becomes a "**Harper lane,**"* with its own SPEC → PLAN → KIT → EVAL → GATE loop, validated through human-in-the-loop governance.



Andrea Franco

November 5, 2025

CLike - **IDEA**: CoffeBuddy an Event-Driven Caffeine

Vision
CoffeeBuddy streamlines office coffee runs entirely within the corporate network: teammates submit orders via Slack, one teammate is fairly assigned as runner, reminders are sent, and preferences are remembered—without relying on public cloud.

Problem Statement
Teams coordinate coffee orders in ad-hoc Slack threads. Messages get buried, someone forgets to pick up, and no one remembers preferences. In regulated environments, external cloud services are restricted, so the solution must run fully on-prem.

Target Users & Context
- Primary: office teammates placing and picking up coffee orders.
- Secondary: office managers seeking fairness and reduced coordination time.
- Context: Enterprise Slack workspace; on-prem Kubernetes; internal identity and gateways only.

Thank You

AI-Native Development • Harper Methodology • Vibe Coding