Sustainable Skills Generate Scalable Agentic Frameworks

dkube.io · Velocity Team · One Convergence

Preprint · v1 · July 2026 · method / position paper

Abstract. Agentic capability today is shipped either as framework-bound code or as opaque prompt bundles: brittle, hard to port, hard to govern, and quick to decay. We argue that the durable unit of agentic capability is a *sustainable skill*—a legible, Markdown-defined capability layered over a thin, deterministic core—and that a catalog of such skills composes into scalable agentic frameworks by *addition* rather than by monolith. We describe a repeatable *pipeline-to-skill* method: elicit an organization's pain points; select features that are sustainable, scalable, and stable; use knowledge-graph memory to build a staged pipeline run self-hosted or in the cloud; harden it against tests; and only then crystallize it into a skill installable with a single command across heterogeneous agent runtimes (Claude Code, and via a portable core, Cursor and Codex). Because the intellectual property is Markdown, the resulting artifact is localizable, forkable, designed for recursive self-improvement, and—over time—native to its environment. We ground the argument in three built instances (an application factory, a data-team pipeline, and a testing harness), one of which we take end-to-end.

Keywords: agentic frameworks, LLM skills, pipeline-to-skill, knowledge-graph memory, data governance, consensus ensembles, sovereign AI, recursive self-improvement.

1 Introduction

Two failure modes dominate how agent capabilities are delivered. The first is *framework lock-in*: capability is written as code against one orchestration framework, so it cannot travel to another runtime and cannot be read by the domain expert who understands the problem. The second is the *opaque prompt*: a monolithic instruction blob that works until it silently does not, with no contracts, no provenance, and no path to improvement. Both decay. Neither scales past the team that authored it.

We take a different position. The durable, composable unit of agentic capability is a *sustainable skill*: a capability whose judgment lives in *legible Markdown* (personas and an orchestrating skill file), whose mechanics live in a *deterministic, dependency-light core*, and whose stages are joined by explicit contracts. Such a skill is portable, governable, and—critically—editable by the people who use it. A library of sustainable skills is a *scalable agentic framework*: it grows by adding legible parts, not by enlarging one system.

Our contributions are (i) a repeatable *pipeline-to-skill* method (§2); (ii) an architecture for sustainable skills and the stage primitives they reuse (§3–4); and (iii) three built instances as evidence, one carried end-to-end (§5). We are explicit throughout about what is demonstrated versus what is architected-for (§7).

2 The Pipeline-to-Skill Method

The method is a loop, not a waterfall (Fig. 1). Its steps:

1) Discover. Work with the organization to surface real pain points, then draw a feature list filtered by three tests—*sustainable* (maintainable by the people who will own it), *scalable* (grows without a rewrite), and *stable* (degrades predictably). Features that fail the filter are deferred, not built.

2) Remember. Decisions, rationale, and constraints are written to a knowledge-graph memory (a linked Markdown vault with typed nodes and back-links). This is what lets the loop *compound*: each pass resumes with full context instead of re-deriving it.

3) Build the pipeline. Implement the capability first as real, staged services—run either self-hosted (sovereign) or in the cloud—so the design is validated against production shapes, not slideware.

4) Test / dogfood. Exercise the pipeline on real data with reconciliation, lineage, and receipts, and gate it behind an automated test suite.

5) Crystallize. Only once the pipeline is proven, extract it into a skill: the reasoning becomes Markdown personas; the mechanics become a small, testable core; the seams become explicit contracts.

6) Distribute. Publish as a one-line install—a plugin marketplace entry and an `npx` launcher—so any user drops it into their agent runtime and invokes it as a slash command.

7) Adapt. Because the artifact is Markdown, adopters localize it, fork it for their domain, and let their agent revise it under test. Over time the skill becomes native to their environment, and its lessons feed the next Discover pass through the same knowledge graph.

3 Anatomy of a Sustainable Skill

A sustainable skill has three layers and two planes (Fig. 2). The **IP layer** is Markdown: a skill file that orchestrates, and one persona per stage that carries the judgment a program cannot. The **core layer** is a small, deterministic, dependency-light program—standard-library only, no virtual environment—so its numbers are reproducible regardless of

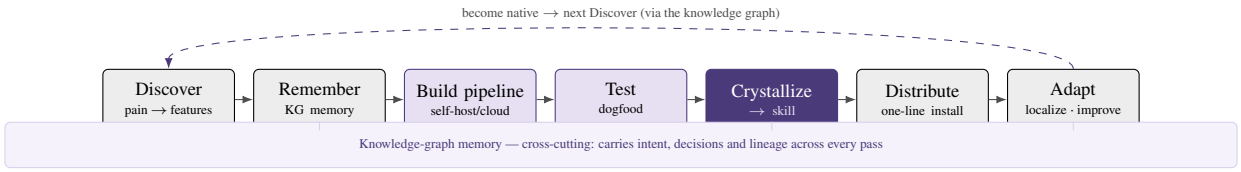

Figure 1: The pipeline-to-skill loop. Capability is proven as a running pipeline before it is crystallized into a distributable skill; a knowledge-graph memory underlies every pass so the method compounds rather than restarts.

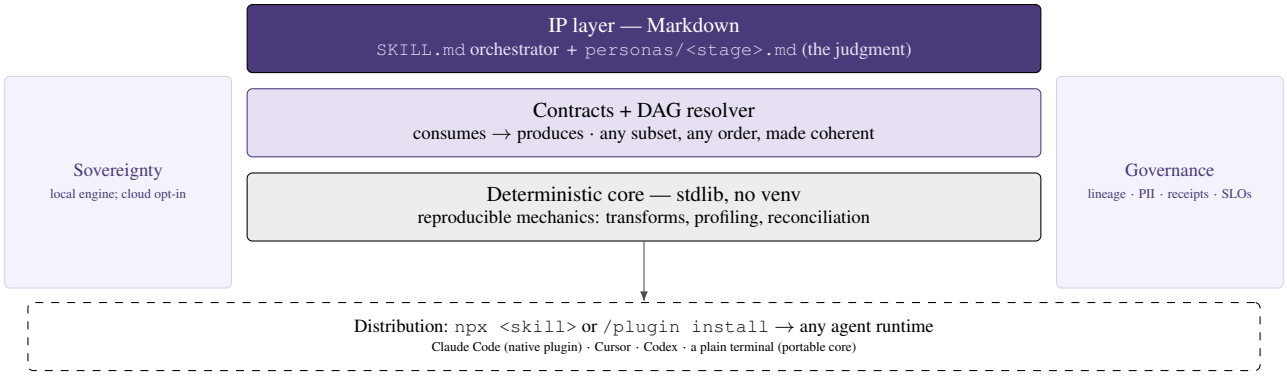

Figure 2: Anatomy of a sustainable skill: legible Markdown IP over a deterministic core, joined by stage contracts, with sovereignty and governance as cross-cutting planes, distributed as a one-line install into any agent runtime.

which model reasons over them. Between them sit **contracts**: each stage declares what it consumes and produces, and a resolver composes any chosen subset into a coherent directed acyclic graph, synthesizing missing upstream stages on demand.

Two planes cut across all stages. *Sovereignty*: the user's local model is the default engine; no data leaves the machine unless a stage is explicitly opted into a cloud model, and mechanical work stays deterministic code. *Governance*: every stage records lineage, flags and handles PII, and emits human-readable receipts, which a final report aggregates into a single verdict. The result is a capability that is correct by construction and auditable end to end, yet still fits in one portable run directory.

4 Reusable Stage Primitives

Across skills, a handful of primitives recur and are worth naming. **Consensus by council**: rather than trust one inference, a heterogeneous panel of models (different families, hence uncorrelated errors) is reconciled into an answer plus a receipt—an *ensemble/council of experts* at the orchestration layer, distinct from an intra-model mixture-of-experts. **Governed definitions**: a metric is defined once, with an additivity rule enforced by a re-aggregation guard so that a ratio is recomputed from components and a distinct-count is routed to a sketch, never silently summed. **PII handling**: sensitive columns are flagged at ingest and their disposition tracked to output. **Reconciliation receipts**: derived facts are proven against their source. These primitives are themselves legible Markdown-plus-core, so they are reused, not re-implemented.

5 Instances

The method is not hypothetical; it produced three skills (Table 1). We take the data-team skill end-to-end as the worked example: eight optional data roles—a spine (architect, engineer, designer, analyst), two branches off the gold layer (BI; scientist → ML), and a platform plane (SRE)—each extracted from a previously running analytics platform. It lands raw files as a typed medallion whose gold measures are reconciled to the raw fact, exposes a governed semantic layer, answers questions with receipts, trains and serves a model, and emits deployment manifests. It is standard-library only, ships with an automated test suite that passes, and is published to a public plugin marketplace and to a package registry with one-line install verified end to end.

6 Why Markdown Is the Substrate

The choice to keep the IP in Markdown is the crux, not an incidental. It makes the artifact **legible** (a domain expert reads and edits it), **portable** (any agent or host runs it), **localizable** (translate the persona), **forkable** (adopters gain an early-mover advantage with no lock-in), and **designed for recursive improvement** (an agent can revise its own personas and re-run the tests). The consequence is adoption by adaptation: rather than forcing an organization onto a fixed tool, the skill bends to the organization until it is native. A skill is thus best understood as a small, governed *Markdown organism* over a reproducible core.

Skill	Shape	Evidence of the method
App Factory	6-stage intent → content pipeline (brain-doc, PRD, build, tests, deploy, marketing)	Model-selectable; personas as Markdown; emits a full product package.
Data Team	8 roles: medallion, semantic layer, analysis, BI/ML/SRE; governance cross-cutting	stdlib-only; gold reconciled to raw fact; tests green; published (plugin + registry); install verified.
TestForge	Multi-dimension testing harness over a target system	Same pipeline-to-skill lineage; reuses governance + receipt primitives.

Table 1: Three instances of the method.

This also reframes upskilling. The scarce resource for the AI era is not only model builders but the far larger population of domain-aware people who can structure and reason about their own data [9]. A sustainable skill is a hands-on, platform-, model-, and hardware-agnostic training substrate: every stage of the data lifecycle is made concrete, so a practitioner learns the whole arc by running it on their data. Not a course about pipelines—the pipeline itself as the classroom.

7 Discussion and Limitations

This is a method paper, and we are careful about its claims. We report *construction and verification* (skills built, tests passing, artifacts published, install verified), not a controlled empirical benchmark; comparative user studies are future work. Recursive self-improvement is *architected-for* (the IP is editable Markdown under test) but is not yet an automated closed loop. "Mixture of experts" in our usage denotes an orchestration-layer council/ensemble [6,7,8], not the sparsely-gated intra-model layer [4]. Finally, native one-click distribution is strongest in runtimes with a plugin system; other runtimes consume the same skill through its portable core and Markdown, which we regard as a feature of the substrate rather than a limitation of it.

8 Related Work

The data stages build on dimensional modeling [1] and the medallion architecture [2], with a governed semantic layer in the lineage of metric layers [3]. The consensus primitive draws on ensemble methods [7], the wisdom of crowds [8], and a society-of-mind view of intelligence [6], and is deliberately distinguished from sparsely-gated mixture-of-experts [4]. The agentic execution model is consonant with reason-and-act prompting [5] and with emerging tool/skill and context protocols for connecting models to external capability [10]. The workforce motivation follows widely cited labor projections [9].

9 Conclusion

Sustainable skills—legible Markdown over a deterministic core, produced by a pipeline-to-skill method and distributed in one line—are the durable, composable unit of agentic capability. The skills are proof; the method is the moat. A catalog of them is a scalable agentic framework that an organization can read, localize, and grow into its own.

References

[1] R. Kimball and M. Ross. *The Data Warehouse Toolkit*, 3rd ed. Wiley, 2013.

[2] Databricks. "What is the medallion lakehouse architecture?" Technical documentation, 2021.

[3] dbt Labs. "The dbt Semantic Layer." Technical documentation, 2023.

[4] N. Shazeer et al. "Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer." *ICLR*, 2017.

[5] S. Yao et al. "ReAct: Synergizing Reasoning and Acting in Language Models." *ICLR*, 2023.

[6] M. Minsky. *The Society of Mind*. Simon & Schuster, 1986.

[7] T. G. Dietterich. "Ensemble Methods in Machine Learning." *Multiple Classifier Systems*, 2000.

[8] J. Surowiecki. *The Wisdom of Crowds*. Doubleday, 2004.

[9] World Economic Forum. "Future of Jobs Report 2023." WEF, 2023.

[10] Anthropic. "Model Context Protocol." Technical documentation, 2024.

Preprint. Corresponding contact: the Velocity Team, dkube.io. The data-team skill is publicly installable. Systems and drafting were developed on the hub2 substrate with the assistance of Claude (Anthropic). This document makes construction-and-verification claims, not benchmark claims.