Claude Code Persistent Memory System

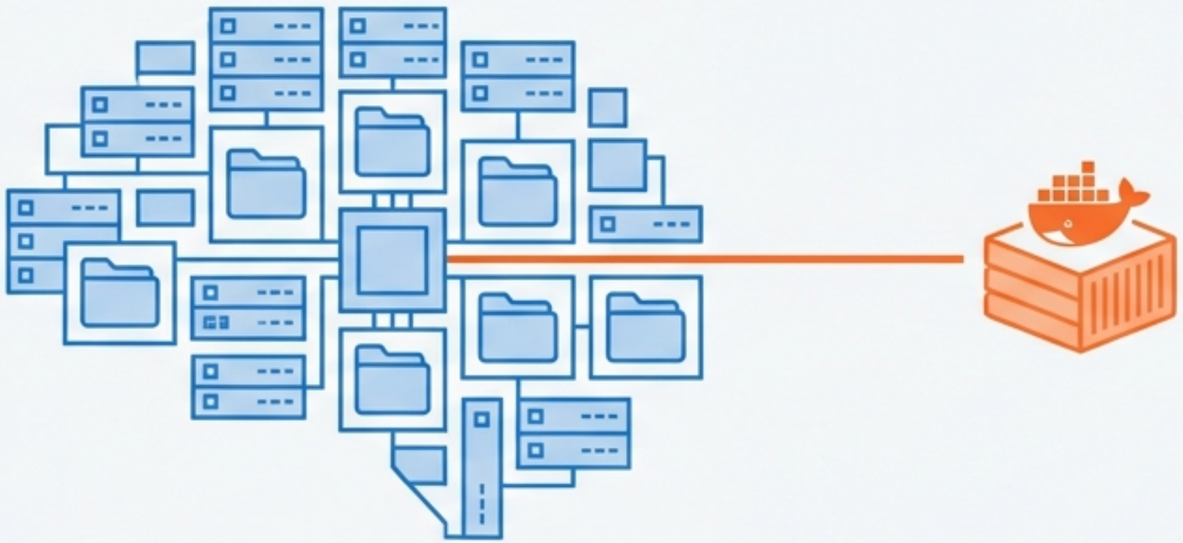
From Stateless Execution to Compounding Intelligence

VERSION: 1.0

DATE: 2026-01-31

STATUS: ACTIVE

DOC TYPE: TECHNICAL OVERVIEW



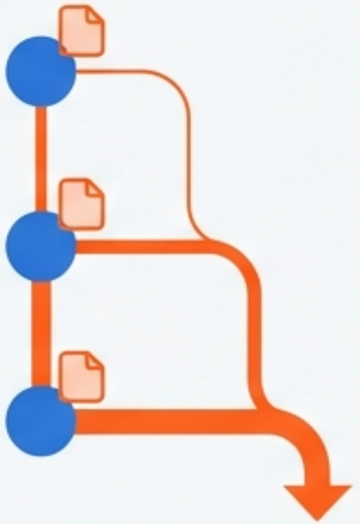
CORE VALUE PROPOSITION: Transform Claude Code from a stateless assistant into a learning system that accumulates knowledge, recognizes patterns, and applies past successes to new tasks.

The Shift to Stateful Intelligence

Stateless / Standard Claude

- Resets every session
- Repeats mistakes
- Zero context retention
- 0% Growth rate

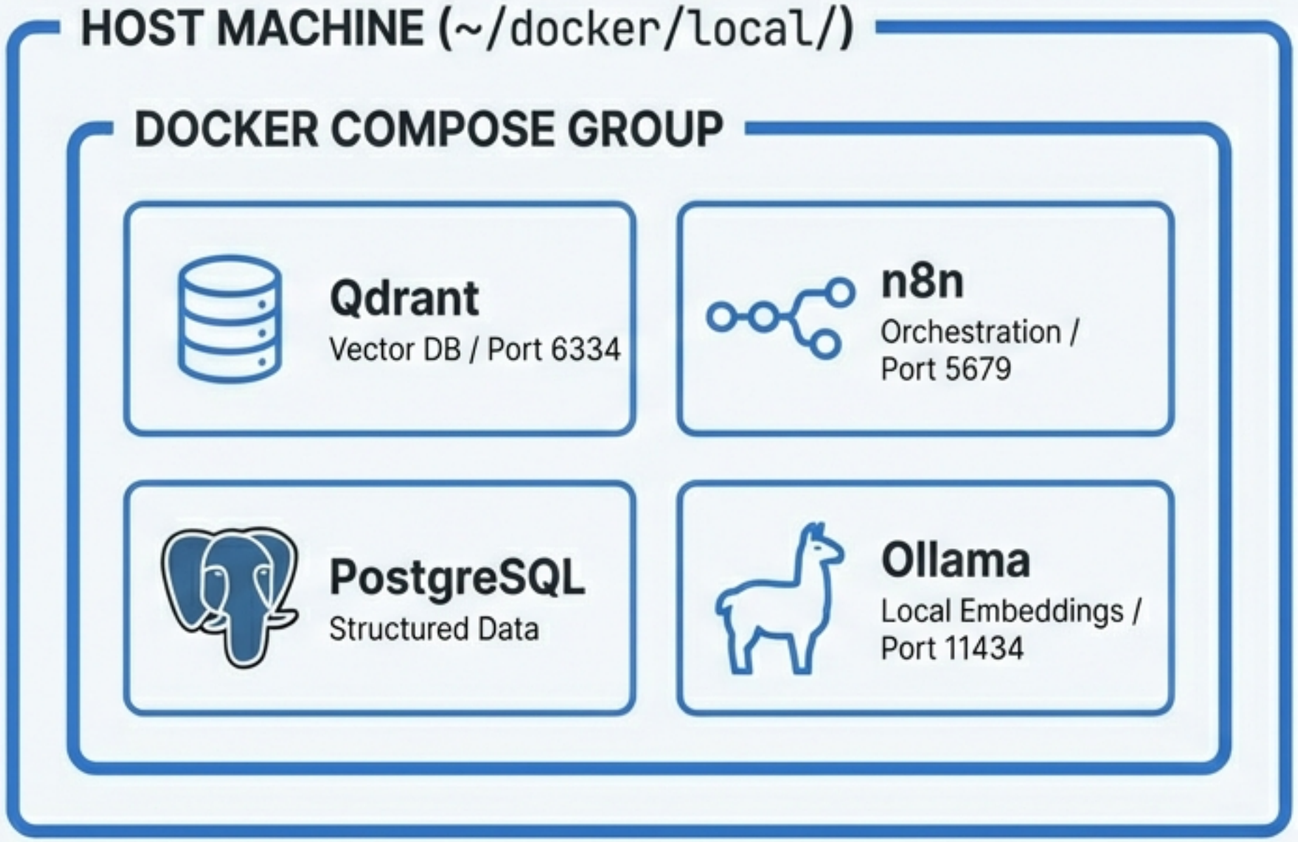
Stateful / Memory-Enabled



- Maintains context
- Learns from execution
- Compounding improvement
- n+1 Efficiency

THE VALUE FORMULA: The more the system is used, the smarter it becomes.
Every success becomes a template for future success.

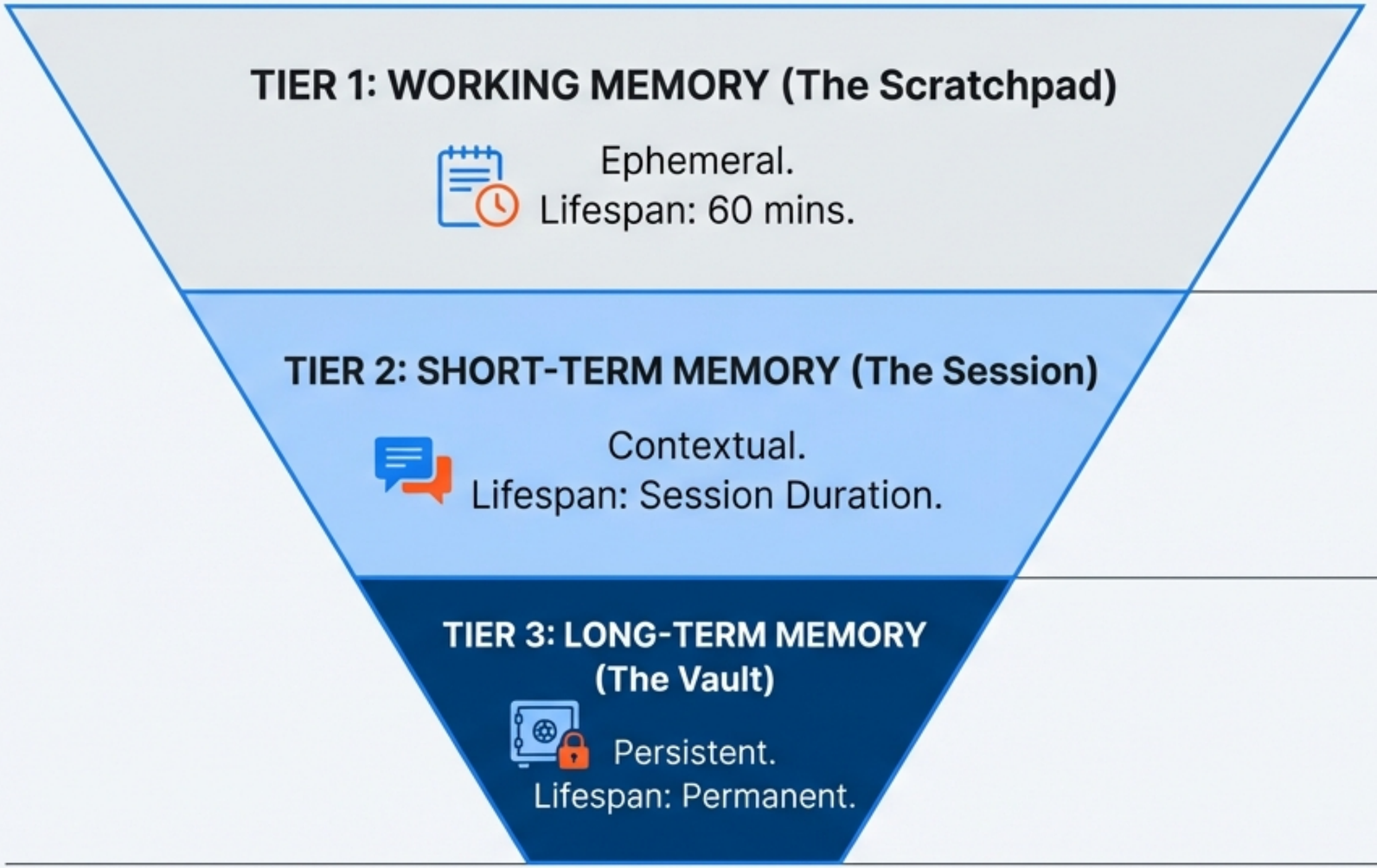
Local Infrastructure & Components



COMPONENT	PURPOSE
Qdrant	Semantic memory storage
n8n	Workflow orchestration
Postgres	Structured storage
Ollama	Host-based embeddings

STARTUP COMMAND: `docker compose up -d` | **HEALTH CHECK:** Verify active ports 6334, 5679, claude-postgres

The Cognitive Tier Architecture



Intermediate reasoning for complex multi-step tasks. Auto-expires.

Persists within a conversation. Promotable to Long-Term.

Stores Facts, Preferences, Decisions, and Context. Survives restart.

Core Memory Operations (API)

memory_store

`params: { content, type, project, tags }`

Store content with semantic embeddings into long-term vault.

memory_recall

`params: { query, limit, include_short_term }`

Search via natural language query across tiers.

memory_scratch

`ops: { create, read, update, task_id }`

Manage the 60-minute ephemeral working memory slot.

memory_promote

`flow: Working -> Short-term -> Long-term`

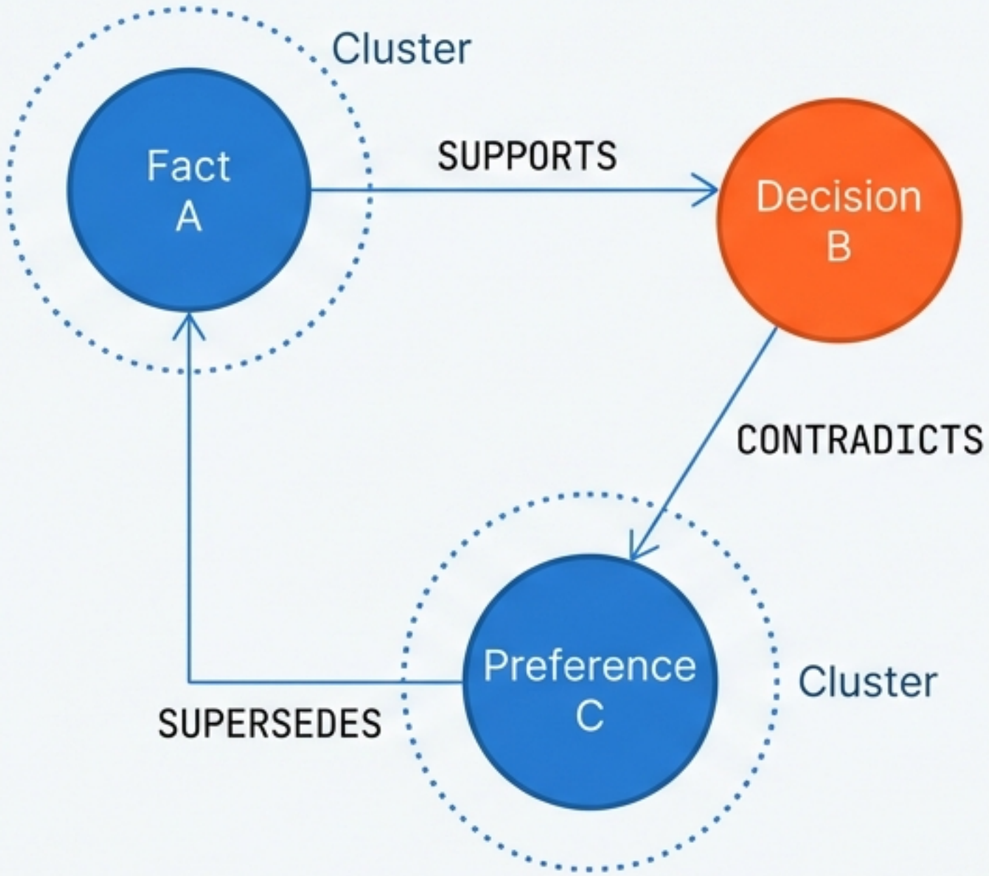
Elevate important context to more permanent tiers.

memory_summarize

`input: { memory_ids[] }`

Consolidate fragmented memories into a single summary.

Organizing the Knowledge Graph

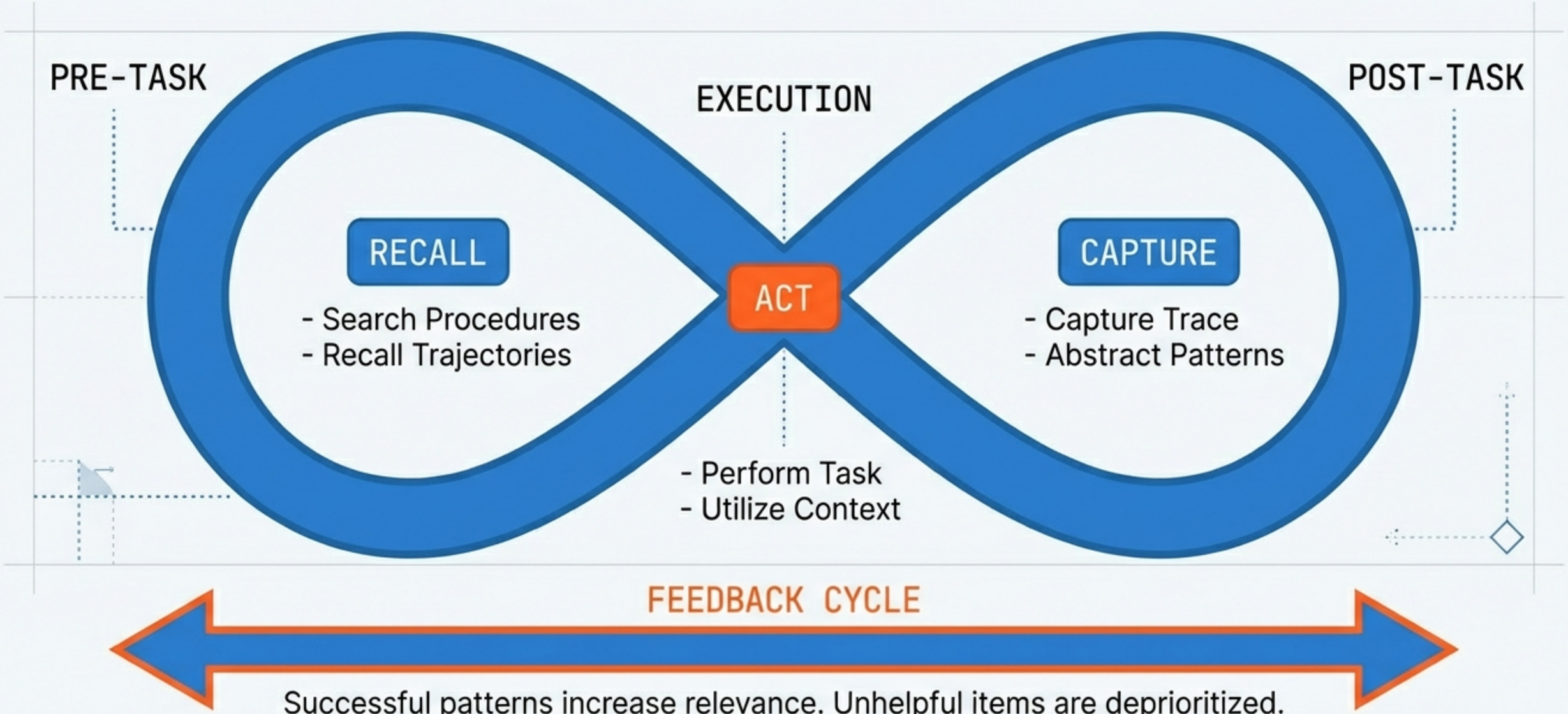


TOOL: memory_organize

- `link / unlink` : Create or remove semantic edges
- `traverse` : Navigate the graph structure
- `cluster` : Group similar vectors (Threshold 0.85)
- `prune` : Remove stale or superseded nodes

RELATIONSHIP TYPES: supports, contradicts, extends, supersedes, related, prerequisite, derived_from

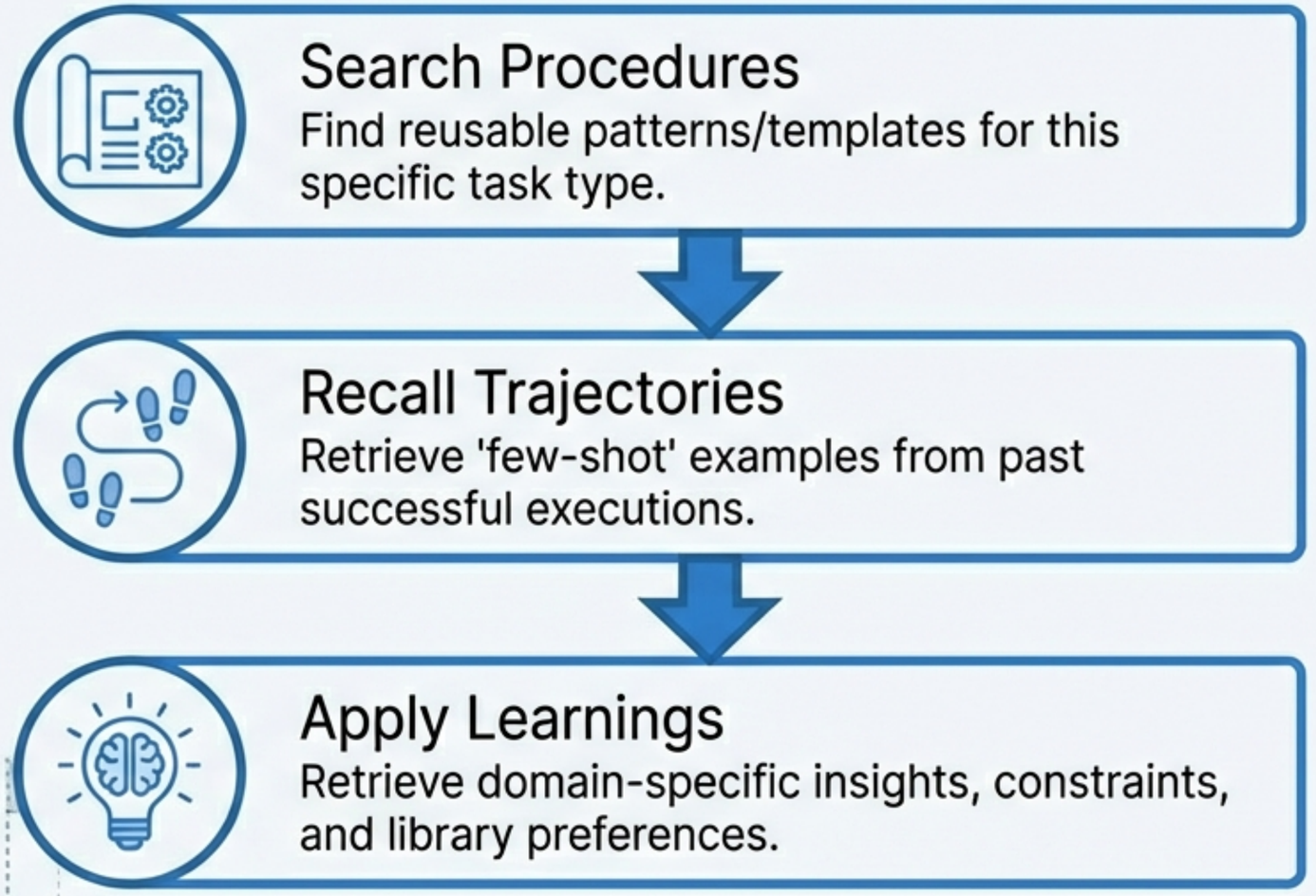
The Compound Intelligence Learning Loop



Successful patterns increase relevance. Unhelpful items are deprioritized.

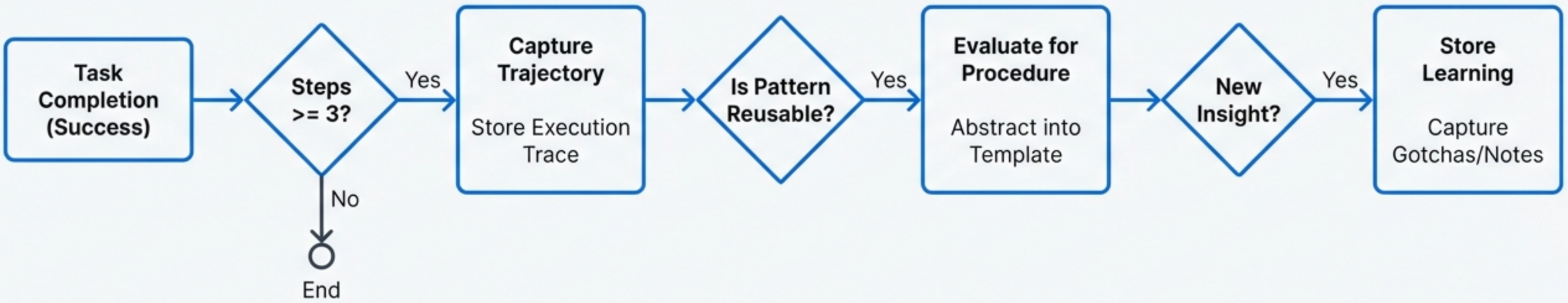
Pre-Task Protocol: Retrieval & Planning

Mandatory sequence triggered at the start of any implementation task.



AUTOMATION TRIGGER:
Initiated automatically when task_type == 'implementation'.

Post-Task Protocol: Capture & Abstraction

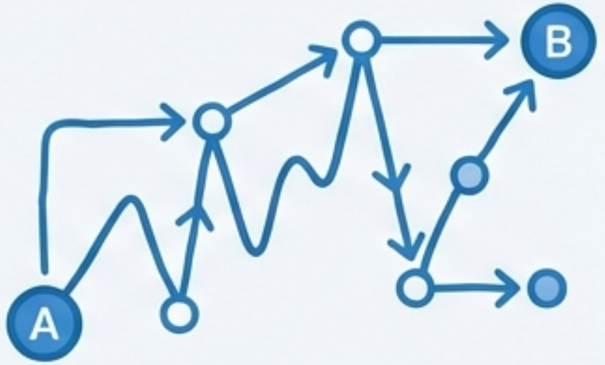


Capture Criteria Checklist

- [x] Step count >= 3
- [x] Likely to recur
- [x] Abstractable (generalizable)
- [x] Novelty (no existing procedure)

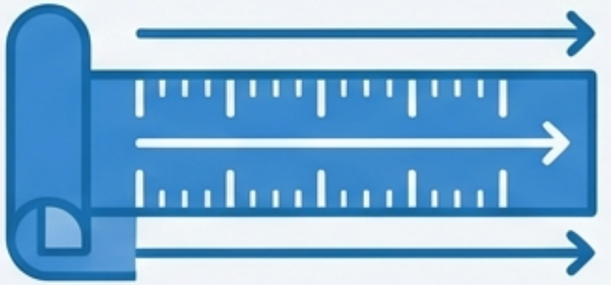
Mechanisms of Experience: Trajectories vs. Procedures

TRAJECTORY (The Specific)



- **Purpose:** Few-shot learning examples.
- **Data:** Step-by-step trace, specific decisions, outcomes.
- **Operation:** `trajectory` (store, recall).
- **Use Case:** "How did I solve this exact error last time?"

PROCEDURE (The General)



- **Purpose:** Generalized templates.
- **Data:** Trigger keywords, template steps, decision logic.
- **Operation:** `procedure` (capture, apply).
- **Use Case:** "What is the standard way to set up a new API endpoint?"

Knowledge Acquisition & Correction

External Knowledge

`rag_search`

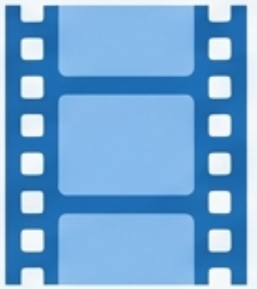
Search Obsidian vault via semantic similarity. Returns document chunks (threshold 0.4).

Internal Wisdom

`learning`

Store insights from tasks. Errors are stored as "anti-patterns" to prevent repetition.

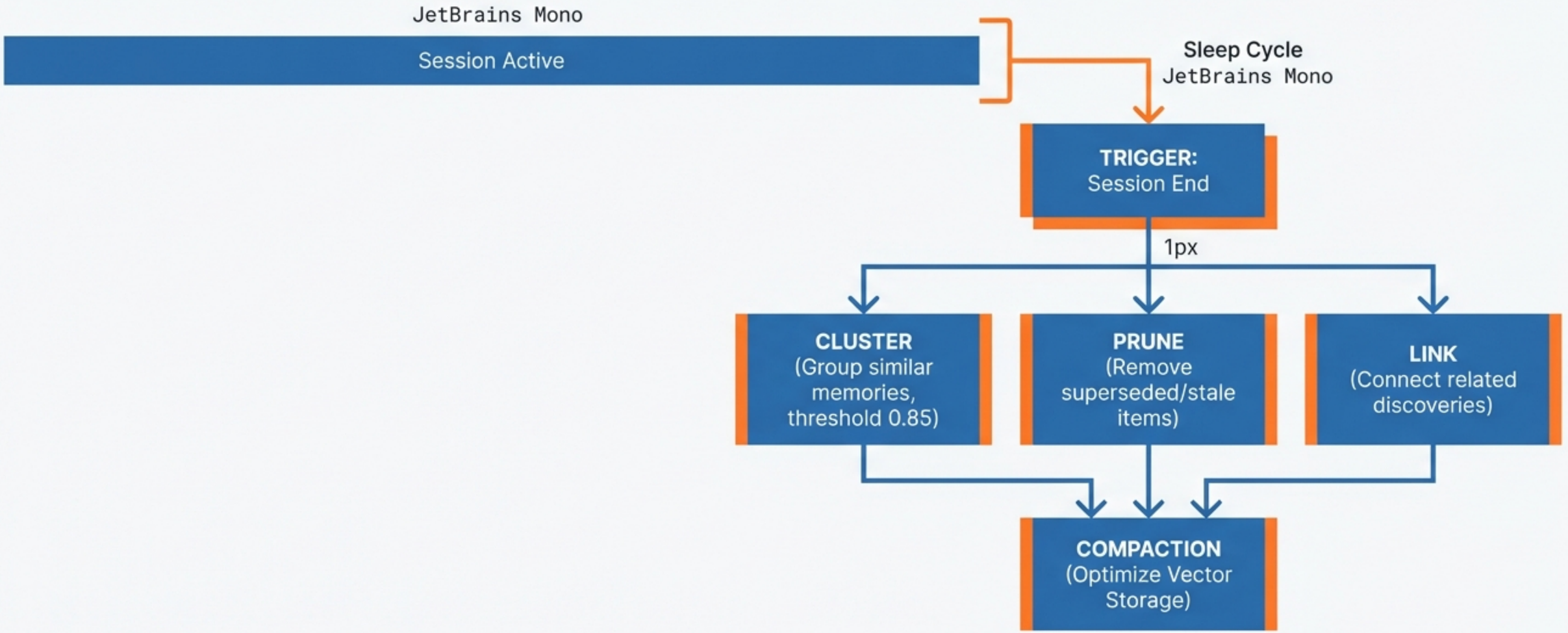
Operational History



`episode`

Record full context: tools used, agents invoked, files modified.

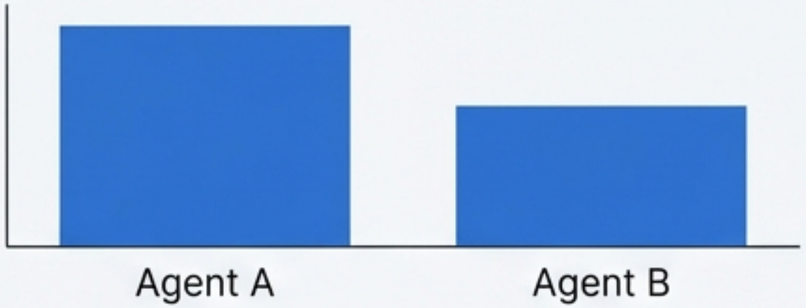
Automated Maintenance & Self-Organization



System autonomously organizes memory during idle time to ensure query efficiency.

Benchmarking & Performance Metrics

Success Rate



Ratio of completed vs. failed tasks.

Efficiency

Demonstrates compounding efficiency.

Speed

Time to complete recursive tasks decreases over time.

TOOL: `benchmark` (record, query, compare)

Security, Privacy & Data Sovereignty

- **All Data Local:** No external API calls for memory.

- **Local Embeddings:** Ollama runs on host machine.

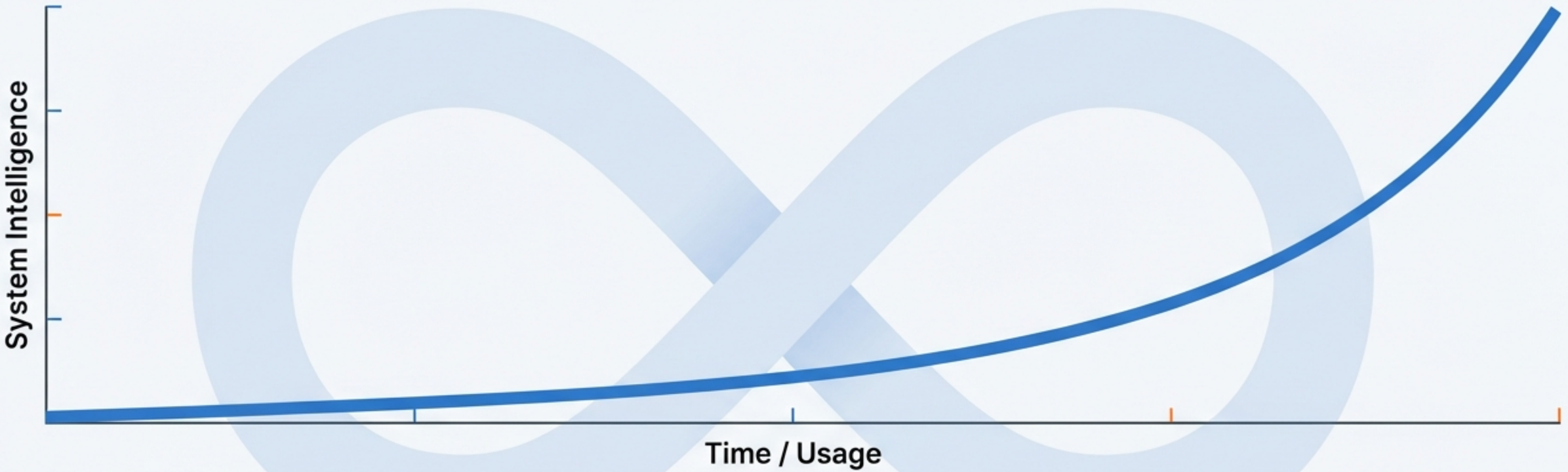
- **No Cloud Dependencies:** Fully self-hosted Docker stack.

- **Safety Protocol:** Credentials/Secrets are explicitly excluded from memory.

Appendix: Data Structure

Qdrant Collections: long_term_memory, short_term_memory, working_memory, episodes, learnings, benchmarks, procedures, trajectories, obsidian_vault

The Compounding Intelligence Future



1. From STATELESS (Amnesia) -> STATEFUL (Wisdom)
2. From MANUAL Repetition -> AUTOMATED Recall

Every success becomes a template. The memory system ensures that we never solve the same problem twice.