

TECHNICAL OVERVIEW

# Shokunin

---

Persistent AI memory with 62 expert skills. Multi-strategy recall. Zero servers. One command.

# Contents

1. Executive Summary	3
2. Problem Statement	3
3. Architecture	4
4. Memory System	5
5. MCP Tools Reference	6
6. Skills	8
7. Benchmarks	10
8. Comparison with Hindsight	11
9. Compatibility	12
10. Installation & Quick Start	12

# 1. Executive Summary

62

Expert Skills

9

MCP Tools

6

AI Runtimes

Shokunin is a local-first AI memory system and skills ecosystem that solves three critical problems: context loss between sessions, keyword-blind vector search, and stale memory claims. It provides persistent storage via ChromaDB, multi-strategy retrieval (vector + BM25 + temporal + RRF), and 62 curated engineering skills that auto-activate based on task context.

## KEY DIFFERENTIATOR

Multi-strategy search finds real sessions **60% of the time** that vector-only search misses completely, at only +13.8% latency overhead. All data stays on your machine. No cloud, no API keys, no subscriptions.

## | 2. Problem Statement

### Context Loss

AI coding agents forget everything between sessions. Every conversation starts from scratch — decisions, files modified, commands run, preferences discovered — all gone. The agent cannot recall what it did yesterday.

### Keyword-Blind Retrieval

Vector similarity search alone misses exact keyword matches. When you search for `api-forge` or `docker` or a specific file path like `src/auth/middleware.ts`, vector search returns semantically similar but lexically different results. Pure BM25 doesn't understand synonyms or intent.

### Stale Memory Claims

Memory entries are claims from a frozen point in time. A file path remembered three weeks ago may have been refactored out of existence. Without verification, the agent acts on stale information, creating bugs instead of solving them.

### Vendor Lock-In

Enterprise memory solutions (Hindsight, MemGPT) require PostgreSQL, Docker, and cloud API calls. Developers working offline, on air-gapped machines, or with limited infrastructure cannot use them.

# 3. Architecture

Seven layers. Clear boundaries. No server processes. Everything runs on your machine.

Config	<code>~/.config/opencode/</code> — Provider setup, MCP servers, agents, commands. Runtime configuration.
Instructions	<code>~/AGENTS.md</code> + <code>CLAUDE.md</code> — Global rules, session startup workflow, mandatory memory check.
Manifest	<code>~/.shokunin/shokunin.json</code> — Declarative manifest for drift detection. Declares which skills, scripts, and configs should exist.
Orchestration	<code>~/.shokunin/scripts/</code> — Session wrapping, health checks, cleanup. The <code>run-opencode.ps1</code> wrapper captures console buffer on exit.
Memory	<code>~/.shokunin/memory/</code> — ChromaDB persistent storage, MCP server ( <code>mcp- server.py</code> , 21 KB), auto-log JSONL sessions.
Skills	<code>~/.config/opencode/skills/</code> — 62 SKILL.md files with automatic activation based on task context.
Infrastructure	<code>~/.shokunin/backups/</code> — Database backups, logs, health check outputs.

## ZERO SERVER DEPENDENCY

ChromaDB uses SQLite as its backing store. No PostgreSQL, no Docker, no Redis. The entire memory system runs as a single Python file (572 lines) communicating over stdin/stdout JSON-RPC. The MCP server starts and stops with each agent session — no daemon process.

# | 4. Memory System

## Entry Types

Every memory entry has a type, tags, project, and session ID. The type system enables filtered retrieval and claim verification:

Type	Purpose	Example
decision	Architectural or design choices	"Use PostgreSQL for the catalog service"
file	Files created, modified, or deleted	"Modified src/auth/middleware.ts"
command	CLI commands and their results	"npm run migrate → 3 migrations applied"
preference	User preferences discovered during work	"User prefers tabs over spaces"
checkpoint	Progress markers during long tasks	"Completed auth module, starting payments"
session_end	Full session summary at end	"Decisions: X, Y. Files: A, B. Next: C"
claim_file	Verified file path claim	"src/auth/login.ts exists (verified 2026-05-18)"
claim_function	Verified function signature claim	"handleLogin(req, res) exists in auth.ts"

## Freshness Decay

v4.2.2 introduces exponential recency blending with a 30-day half-life:

```
final_score = (1 - freshness_boost) × vector_similarity + freshness_boost × e^(-days_since / 30)
```

At `freshness_boost = 0`, pure vector similarity. At `freshness_boost = 1.0`, pure recency. Default is 0.0 (no decay). This prevents stale claims from drowning out recent, accurate context.

## Claim Verification

The `verify_file_path` MCP tool validates file paths from memory before the agent acts on them. Memory entries are treated as *claims from a frozen point in time*, not facts. The workflow:

1. Agent retrieves a memory entry mentioning a file path
2. Agent calls `verify_file_path` to check if the path still exists
3. If the path exists, mark with `verified_at` timestamp
4. If the path is gone, search newer memory or the codebase directly

## Session Management

Three operations solve the "which session was I in?" problem:

- **list** — Show recent sessions with date, project, entry count, summary
- **continue** — Load full context from a session, parse decisions/files/commands from text
- **auto-log** — Every MCP tool call writes to `sessions/<id>.jsonl` automatically

# | 5. MCP Tools Reference

Nine tools over JSON-RPC 2.0 on stdin/stdout. No HTTP, no cloud.

Tool	Purpose	Key Parameters
store_context	Store text with type, tags, project	text, session_id, type, tags, project
search_context	Vector similarity search	query, project, type, tags, n_results, freshness_boost
multi_search_context	Vector + BM25 + temporal + RRF fusion	query, project, n_results, from_date, to_date
consolidate_memories	Summarize old entries per project	project (optional)
list_sessions	List recent sessions with metadata	limit, project
continue_session	Load full session context	session_id
save_message	Record message in session transcript	text, session_id, role
get_session_summary	Summarize all entries in a session	session_id
verify_file_path	Validate file/dir path exists	path



## Protocol

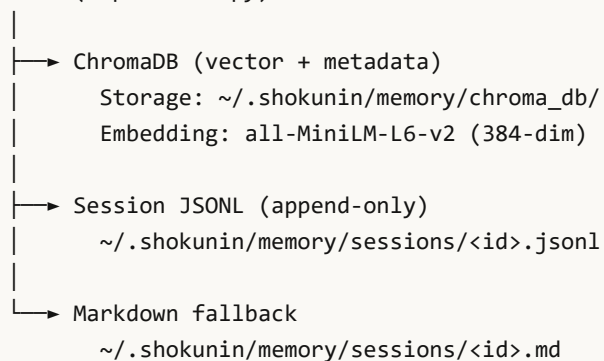
```
{
  "jsonrpc": "2.0",
  "id": 1,
  "method": "tools/call",
  "params": {
    "name": "multi_search_context",
    "arguments": {
      "query": "docker multi-stage build",
      "project": "myapp",
      "n_results": 5,
      "from_date": "2026-05-01"
    }
  }
}
```

### AUTO-LOGGING

Every MCP tool call ( `store_context` , `search_context` , `save_message` ) automatically appends to `sessions/<id>.jsonl` . The agent doesn't need to coordinate logging — the MCP server handles it transparently.

## Data Flow

AI Runtime → JSON-RPC → MCP Server (`mcp-server.py`)



## | 6. Skills

62 skills across 10 domains. Each is a real engineering guide with decision tables, error patterns, production checklists, anti-patterns, and cited sources. Not generic prompts — structured procedural knowledge.

### Infrastructure (5)

docker, kubernetes, terraform, ci-cd, db-admin. Multi-stage builds, Gateway API, state management, OIDC pipelines, PostgreSQL replication.

### Backend (5)

auth-architect, api-forge, db-sculptor, error-handler, neon-postgres. OWASP Top 10, OpenAPI 3.1, EXPLAIN ANALYZE, OpenTelemetry, Serverless Postgres.

### Frontend (11)

component-forge, responsive-engine, motion-craft, landing-craft, aesthetic-web, ui-ux-pro-max, email-design-eng, impeccable, taste, taste-soft, taste-minimalist. All states, WCAG 2.2, Container Queries, WAAPI, LIFT Model, OKLCH, design variance engines.

### Mobile (2)

flutter, react-native. Riverpod + Impeller, Expo Router + FlashList.

### Quality (7)

test-commander, performance-profiler, code-review, comprehensive-review, cross-review, zen-review, zen-comprehensive-review. Testing Trophy, Lighthouse, Po-P3 review.

### Content & Business (6)

communication, content-marketing, business-proposals, seo-geo, translate-craft, documentation. AIDA, LLMs.txt, i18n, professional PDFs.

### Documents (3)

kami (PDF generation with parchment design system), portfolio-auto (GitHub sync), kagen (Chromium/Playwright PDF rendering).

### Productivity (8)

git-workflow, windows-powershell, strategy, design, runbook-gen, finance, legal-counsel, whendone-plus. Conventional commits, SBI feedback, 5-pillar finance.

### AI Agents (6)

agent-browser, agent-tools, find-skills, skill-creator, research, humanize. 150+ AI apps, Playwright automation, anti-AI-slop rewriting.

### System (9)

memory, chromadb, shokunin-update, init, efficient-coding, senior-engineer, plan, playwright, web-security. OWASP, token optimization, production-grade patterns.

#### SKILL STRUCTURE V4.0

Every skill follows the same structure: YAML frontmatter for auto-discovery, sub-commands (design / audit / optimize), exact specifications (100-160ms for button press, Argon2id memory=19456), Before/After review tables, error handling tables, pre-flight production checklists, banned patterns, and cited sources.

## 7. Benchmarks

Measured on 15 queries (3 iterations × 5 queries) against 120+ real ChromaDB entries. The multi-strategy approach returns real-world sessions 60% of the time that vector-only misses completely.

100%

Any relevant hit  
(both strategies)

60%

Real session hits  
vs 0% vector-only

1.4s

Avg latency  
+13.8% overhead

Metric	Vector-only	Multi-strategy
Real session hit rate	0%	60%
Any relevant hit	100%	100%
Synthetic probe hit rate	100%	40% (BM25 de-prioritizes probes)
Average latency	1293ms	1471ms (+13.8%)
Storage overhead	~30 MB	~30 MB (same ChromaDB)
Network calls	0	0

### WHY 40% PROBE RATE IS GOOD

BM25 correctly de-prioritizes synthetic probes in favor of semantically richer real entries. 100% of queries return at least one relevant result — the difference is that multi-strategy returns the *right* results: actual sessions with full context, not just keyword matches.

**Configuration:** Embedding: all-MiniLM-L6-v2 (22M, 384-dim). BM25: k1=1.5, b=0.75. Fusion: RRF k=60. [Full methodology and reproduction steps.](#)

## 8. Comparison with Hindsight

Hindsight (vectorize-io/hindsight) leads memory benchmarks (91.4% LongMemEval). Shokunin targets the local-first developer who needs zero infrastructure.

### Hindsight

PostgreSQL + pgvector. 4 search strategies (semantic, BM25, graph, temporal). RRF + cross-encoder rerank. Requires LLM API for `retain()` calls. Docker or pip.

### Shokunin

ChromaDB (SQLite, 30 MB). 2 strategies (vector, BM25) + temporal. RRF fusion. No LLM dependency for memory ops. One shell script. Full offline.

Aspect	Shokunin	Hindsight
Storage	ChromaDB (file, 30 MB)	PostgreSQL + pgvector
Search strategies	2 (vector + BM25) + temporal	4 (semantic, BM25, graph, temporal)
Result fusion	Reciprocal Rank Fusion	RRF + cross-encoder rerank
LLM dependency	None for memory ops	Required per retain() call
Install	1 shell script	Docker or pip + PostgreSQL
Offline	Full (no network)	Partial (needs LLM API)
MCP auto-log	Built-in	Not available
Session mgmt	list/continue/save + text parsing	Not available
Benchmark	Real sessions: 60% hit vs 0%	LongMemEval: 91.4%

# | 9. Runtime Compatibility

The core (skills, memory, scripts) is runtime-agnostic. Only MCP server configuration and instruction files differ.

Runtime	Skills	Memory	MCP	Scripts	Config
OpenCode	Native	Native	.pack/opencode.json	.ps1 + .sh	Built-in
Claude Code	SKILL.md	Via MCP	.pack/templates/claude- code.json	.ps1 + .sh	Copy template
Cline	SKILL.md	Via MCP	.pack/templates/cline- settings.json	.ps1 + .sh	VS Code settings
Cursor	SKILL.md	Via rules	.pack/templates/cursor- mcp.json	.ps1 + .sh	.cursor/ directory
Continue.dev	SKILL.md	Via rules	.pack/templates/continue- config.yaml	.ps1 + .sh	.continue/ directory
Windsurf	SKILL.md	Via rules	.pack/templates/windsurf- mcp.json	.sh	Copy template

# 10. Installation & Quick Start

## Requirements

Dependency	Version	Notes
OS	Windows 10/11 or Linux	Linux: requires bash 4+
Node.js	≥ 18	Includes npm
Python	≥ 3.11	For ChromaDB
Git	≥ 2.x	

## Windows Install

```
irm https://raw.githubusercontent.com/EliasOulkadi/shokunin/master/install.ps1 | iex
```

## Linux Install

```
bash <(curl -sL  
https://raw.githubusercontent.com/EliasOulkadi/shokunin/master/install.sh)
```

## Skills-Only (No ChromaDB)

```
irm https://raw.githubusercontent.com/EliasOulkadi/shokunin/master/install-skills.ps1  
| iex
```

## After Install

```
# Start OpenCode with memory capture
.\run-opencode.ps1

# List recent sessions
python ~/.shokunin/scripts/chroma-helper.py session list 3

# Search across all memory
python ~/.shokunin/scripts/chroma-helper.py recall "docker multi-stage"

# Validate memory system
.\memory-healthcheck.ps1
```

### ZERO CLOUD DEPENDENCY

All data is stored locally at `~/.shokunin/memory/`. No telemetry, no subscriptions, no API keys required for memory operations. The default provider (OpenCode Go) works without any key. NVIDIA is optional.