kg

Knowledge Graph

Reference Documentation

JSONL

Source of Truth

SQLite

Derived Index

MCP

LLM Integration

Lightweight · Git-friendly · Hybrid Search · Local-first

Contents

Introduction

kg is a lightweight, file-first knowledge graph designed for software projects and AI-assisted workflows. It keeps knowledge as human-readable JSONL files that are trivially grep-able, diff-able, and git-trackable, while deriving a SQLite index for fast hybrid search.

> **Design Philosophy**
>
> **Files are the source of truth.** SQLite is a derived cache you can always rebuild with `kg reindex`. The graph survives any database corruption — your notes never will.

Why kg?

Traditional note systems force you to choose between structure (databases, heavy editors) and simplicity (plain text). kg offers a third path:

Plain text wins for durability

- `jq`, `rg`, `git log` work natively
- Merge conflicts are readable
- No lock-in: export is just `cat`
- Works offline, always

Index wins for retrieval

- FTS5 BM25 keyword search
- Vector similarity (local or cloud)
- Calibrated hybrid score fusion
- Cross-encoder reranking

Key Concepts

A **node** groups related **bullets** under a **slug**. Think of nodes as Zettelkasten cards and bullets as atomic facts on those cards.

```
project/
  .kg/
    nodes/
      my-topic/
        node.jsonl    ← bullets (append-only log, git-tracked)
        meta.jsonl    ← votes, usage counts (git-tracked)
    index/
      graph.db        ← SQLite: FTS5 + embeddings + backlinks (gitignored)
  kg.toml             ← project config
```

> ⚠ **graph.db is gitignored**
>
> Never commit `graph.db`. It is always derivable from the JSONL files via `kg reindex`. The `.kg/nodes/` directory is what you version-control.

Architecture

Component Overview

The diagram below shows the main components and data flows in kg.

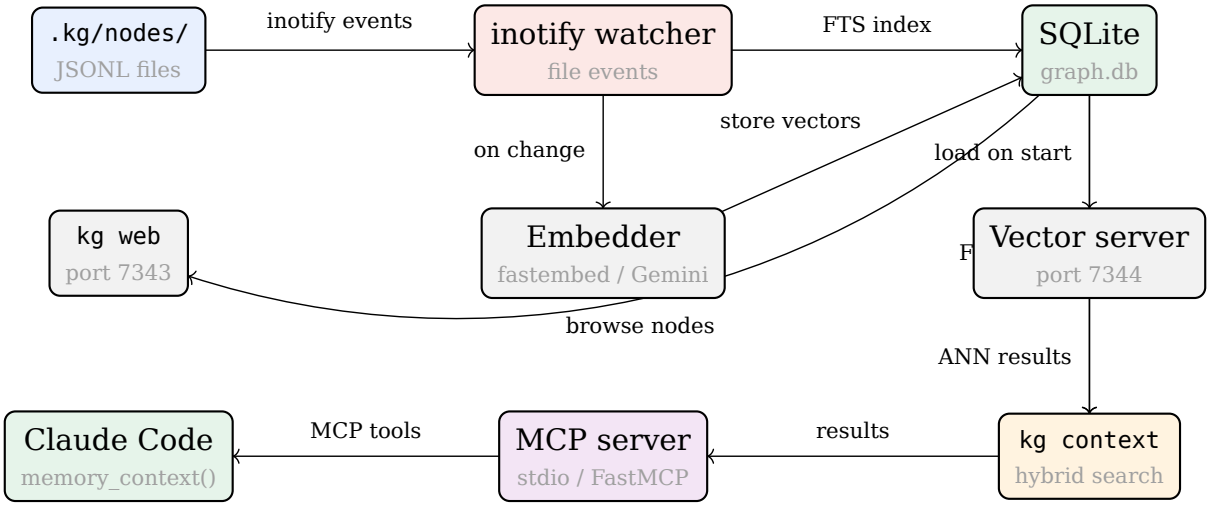

Figure 1: kg component architecture and data flows

Write Path

When you run `kg add my-topic "some fact"`:

1. The CLI appends a new bullet (JSONL line with stable UUID) to `.kg/nodes/my-topic/node.jsonl`, creating the directory if needed.
2. The inotify watcher detects the file change within milliseconds.
3. The watcher calls the indexer, which updates the FTS5 table in SQLite.
4. If embeddings are configured, the embedder generates a vector and stores it in the `embeddings` table.
5. Backlinks (`[[other-slug]]` references) are extracted and written to the `backlinks` table.

> 🔥 **Only the watcher writes to SQLite**
>
> CLI commands like `kg add`, `kg update`, `kg delete` only write JSONL. The watcher is the **sole** writer to `graph.db`. This eliminates concurrent-write corruption.

Read Path

```
kg context "query"
  │
  ├─ FTS search  ──────────────────────┐
```

```
  │    SQLite FTS5 BM25 ranking            │
  │    OR-expanded + prefix wildcards      │  calibrated
  │                                        ├─ score fusion ──→ reranker ──→
output
  ├─ Vector search ────────────────────────┘
  │    cosine similarity via vector server │
  │    nearest-neighbor over embeddings    │
  │
  └─ Session dedup: skip already-seen bullets
```

Storage Layout

Path	Format	Purpose
`kg.toml`	TOML	Project configuration (name, embeddings, sources)
`.kg/nodes/<slug>/node.jsonl`	JSONL	Bullets — source of truth, git-tracked
`.kg/nodes/<slug>/meta.jsonl`	JSONL	Vote counts, usage stats (git-tracked)
`.kg/index/graph.db`	SQLite	FTS5 index, embeddings, backlinks (gitignored)
`.kg/skills/`	Markdown	Claude Code skills, via `.claude→.kg` symlink
`~/.cache/kg/embeddings/`	diskcache	Embedding vector cache (cross-project)

Table 1: File storage layout

Installation

From Git (Latest)

```bash
uv tool install "git+https://github.com/sasha-s/kg.git"
```

With optional extras for embedding support and live file watching:

```bash
uv tool install "kg[embeddings,watch] @ git+https://github.com/sasha-s/kg.git"
```

Development / Editable

```bash
git clone https://github.com/sasha-s/kg.git
uv tool install --editable ./kg
```

Optional Extras

Extra	Packages	Use
`embeddings`	`fastembed`, `google-genai`, `diskcache`, `numpy`	Local + cloud embeddings
`watch`	`inotify-simple`	Live file watching (Linux)
`turso`	`libsql`	Turso remote SQLite (requires cmake)
`dev`	`ruff`, `basedpyright`, `pytest`	Development tools

Table 2: Optional install extras

Quickstart

Initialize a Project

```bash
# In your project directory:
kg init             # writes kg.toml, creates .kg/
kg start            # index + calibrate + watcher + vector server + MCP + hooks
```

After `kg start`, the `.claude → .kg` symlink is created so Claude Code can discover the MCP server and local skills automatically.

First Notes

```bash
kg add my-topic "discovered: X is faster than Y in this benchmark"
kg add my-topic "gotcha: Y breaks when input is empty" --type gotcha
kg add my-topic "decision: use X for production" --type decision
```

Search

```bash
kg search "fast benchmark"        # FTS keyword search
kg context "which is faster"      # hybrid FTS + vector, calibrated + reranked
```

CLI Reference

Core Commands

Command	Description
`kg init`	Create `kg.toml` and `.kg/` directory structure
`kg start`	Index + calibrate + start watcher + vector server + MCP + install hooks
`kg stop`	Stop watcher and vector server
`kg reload`	Hot-reload `kg.toml` config (sends SIGHUP to watcher, no restart)
`kg reindex`	Rebuild SQLite from all `node.jsonl` files (stops/restarts watcher)
`kg calibrate`	Compute FTS/vector score quantiles for hybrid fusion
`kg upgrade`	Run schema migrations + reindex (safe to run anytime)
`kg status`	Show node counts, calibration state, watcher, vector server, config
`kg serve`	Start MCP server on stdio (used by Claude Code)
`kg web`	Start web viewer at `http://localhost:7343`

Table 3: Core management commands

Node Commands

Command	Description
`kg add <slug> <text> [--type TYPE]`	Add a bullet to a node (auto-creates node if missing)
`kg create <slug> <title> [--type TYPE]`	Create a node with explicit title (idempotent)
`kg show <slug>`	Print all bullets in a node
`kg update <bullet-id> <text>`	Update bullet text by ID
`kg delete <bullet-id>`	Delete a bullet by ID
`kg nodes [PATTERN]`	List nodes (glob on slug, e.g. `notes-*`)
`kg nodes --bullets`	Sort nodes by bullet count descending
`kg nodes --recent`	Sort nodes by most-recently-updated
`kg nodes --docs`	Show `_doc-*` source-file nodes instead of curated nodes

Table 4: Node management commands

The `--type` flag accepts: `fact`, `gotcha`, `decision`, `task`, `note`, `success`, `failure`.

Search & Retrieval

Command	Description
`kg search <query> [-n N]`	FTS keyword search, returns raw results
`kg context <query> [-s SESSION_ID]`	Hybrid search + rerank, formatted for LLM injection
`kg context <query> -q <intent>`	Use a separate query string for the reranking step

Table 5: Search and retrieval commands

Review & Quality

Command	Description
`kg review`	List nodes ordered by credits-per-bullet (review debt)
`kg review <slug>`	Mark node as reviewed (clears token budget to 0)
`kg vote useful <bullet-id>...`	Signal that a bullet is high quality
`kg vote harmful <bullet-id>...`	Signal that a bullet is wrong or misleading

Table 6: Review and quality commands

Search & Context

The Hybrid Search Pipeline

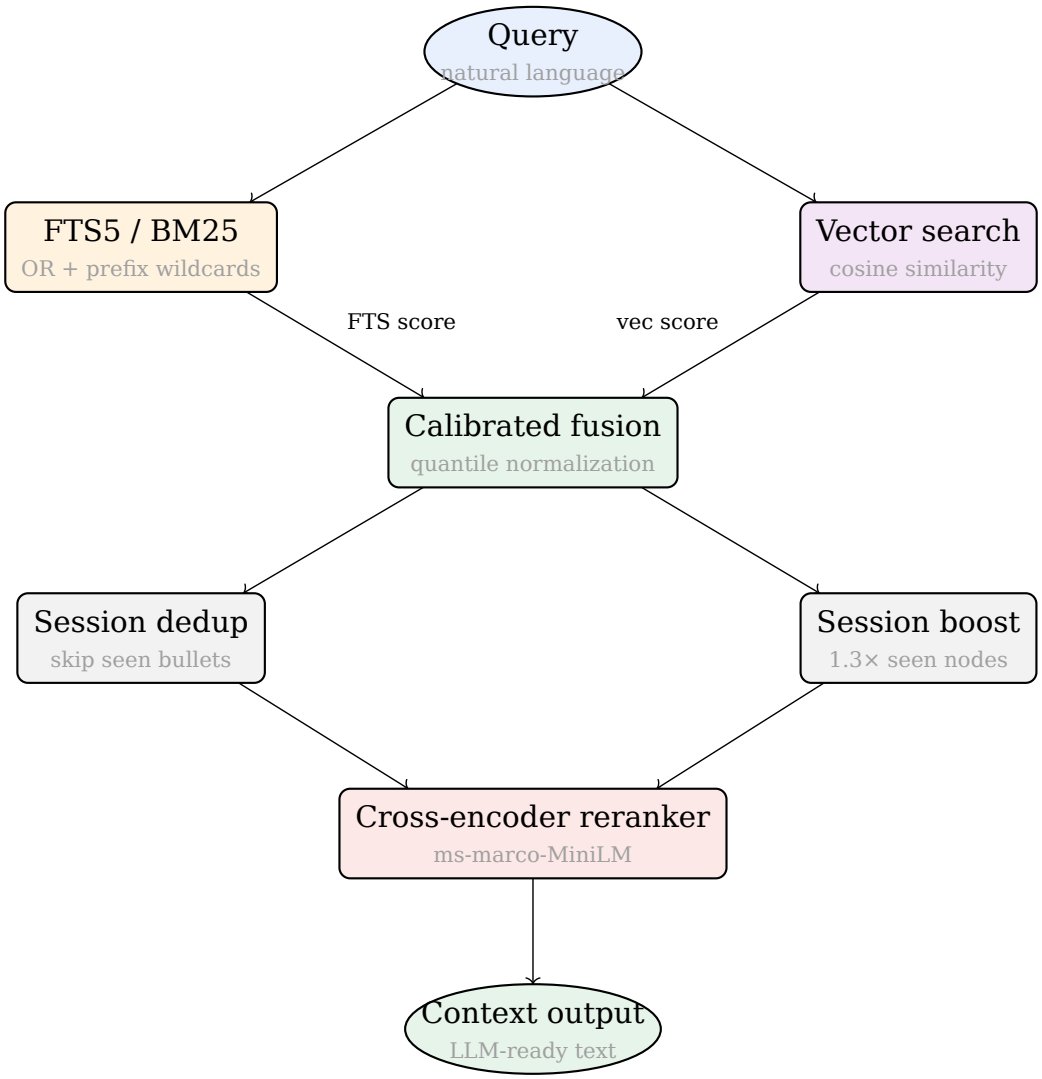

Figure 2: Hybrid search and context retrieval pipeline

FTS Search

SQLite FTS5 BM25 ranking with automatic query expansion:

- Each search term becomes `term OR term*` (prefix wildcard for partial matches)
- Multiple terms are OR-joined for broader recall
- Weight controlled by `fts_weight` in `kg.toml` (default 0.5)

Vector Search

A lightweight HTTP vector server (port 7344) loads embeddings from SQLite at startup and serves approximate nearest-neighbor queries.

Prefix in kg.toml	Provider	Notes
`gemini:...`	Google Gemini	Cloud — requires `GEMINI_API_KEY`
`openai:...`	OpenAI	Cloud — requires `OPENAI_API_KEY`
`fastembed:...`	fastembed	Local, no API key — runs on-device
(bare model name)	fastembed	e.g. `BAAI/bge-small-en-v1.5`

Table 7: Supported embedding providers

Score Calibration

Raw FTS and vector scores are incomparable across queries and models. calibration maps them to percentile quantiles before fusion:

1. Sample 200 random nodes from the graph
2. Run FTS and vector search against each sample
3. Compute breakpoints at p0, p10, p25, p50, p75, p90, p100
4. At query time, binary-search each raw score into its quantile bucket
5. Fuse: `score = fts_weight × fts_q + vec_weight × vec_q + dual_match_bonus`

> 🔥 **Tip**
>
> Run `kg calibrate` after adding many new nodes or after changing your embedding model. The watcher auto-recalibrates when more than `auto_calibrate_threshold` (5% by default) of bullets change.

Cross-Encoder Reranking

After fusion, the top-N candidates are passed to a cross-encoder reranker. The reranker sees the full `(query, bullet)` pair — capturing semantic nuance that cosine similarity misses.

```toml
[search]
use_reranker = true
reranker_model = "Xenova/ms-marco-MiniLM-L-6-v2"
```

Configuration

`kg.toml` lives in your project root. All sections are optional.

```toml
[kg]
name = "my-project"

[embeddings]
model = "gemini:gemini-embedding-001"
# model = "fastembed:BAAI/bge-small-en-v1.5"   # local alternative

[search]
fts_weight = 0.5
vector_weight = 0.5
dual_match_bonus = 0.1
use_reranker = true
reranker_model = "Xenova/ms-marco-MiniLM-L-6-v2"
auto_calibrate_threshold = 0.05  # recalibrate when 5% of bullets change

[review]
budget_threshold = 3000   # credits-per-bullet before review flag (~15 serves)

[server]
port = 7343         # web viewer
vector_port = 7344  # vector server

[[sources]]
name = "workspace"
path = "."
include = ["**/*.py", "**/*.md"]
use_git = true       # only index git-tracked files
```

API secrets in `.env` (gitignored by `kg init`):

```bash
GEMINI_API_KEY=...
TURSO_URL=libsql://...
TURSO_TOKEN=...
```

Source File Indexing

The `[[sources]]` section indexes source files into the same FTS index as curated nodes. Content is automatically chunked (avg 1500 bytes) without LLM extraction.

```toml
[[sources]]
name = "codebase"
```

```toml
path = "/path/to/project"
include = ["**/*.py", "**/*.ts", "**/*.md"]
exclude = [".kg/**", "node_modules/**"]
use_git = true
```

> ⚠ **fnmatch differences: Python 3.11 vs 3.12**
>
> In Python 3.11, `fnmatch` treats `*` as matching `/` — so `*.py` matches `src/main.py`. In Python 3.12+, `*` does **not** match `/`. Use `**/*.py` patterns for reliability across Python versions.

Indexed files appear as `_doc-*` nodes in SQLite, browsable via `kg nodes --docs` and the web viewer.

Review & Token Budget

The Review System

Every time a node contributes bullets to `kg context` output, the character count is added to that node's **token budget**. When `budget ÷ bullet_count` exceeds `budget_threshold` (default 3000), the node is flagged for review.

Dividing by bullet count means every node — large or small — is held to the same standard.

```bash
kg review              # list flagged nodes, ordered by credits-per-bullet
kg review <slug>       # inspect flagged bullets + mark as reviewed (clears budget)
```

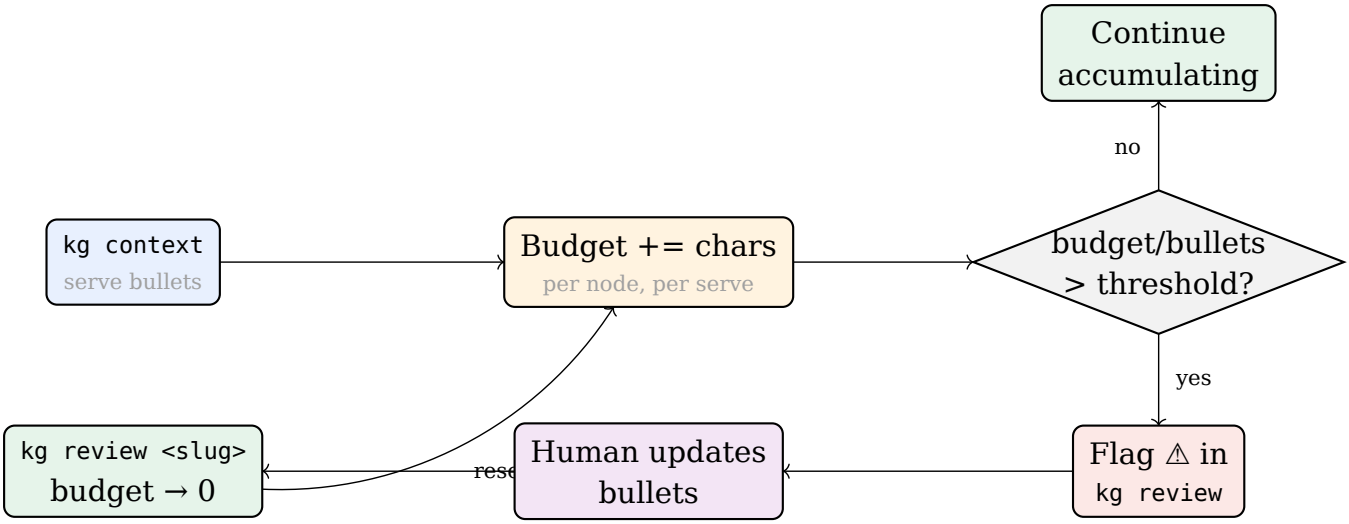

Figure 3: Token budget lifecycle and review loop

Vote Quality Signals

Bullets can be voted on to signal quality. A bullet with more harmful than useful votes is shown with a ⚠ prefix in `kg show` output.

```bash
kg vote useful  <bullet-id>    # signal high quality
kg vote harmful <bullet-id>    # signal wrong or misleading
```

`kg review` surfaces nodes with net-harmful bullets even if under the budget threshold.

MCP Server & Claude Code

Overview

kg implements the Model Context Protocol (MCP) so any MCP-compatible LLM client can use it as a knowledge tool.

```bash
kg start   # registers MCP automatically in ~/.claude/settings.json
```

Or add manually to `~/.claude/settings.json`:

```json
{
  "mcpServers": {
    "kg": {
      "command": "kg",
      "args": ["serve"],
      "cwd": "/path/to/your/project"
    }
  }
}
```

MCP Tools

Tool	Description
`memory_context(query, session_id?)`	Hybrid search + rerank, auto-dedup within session, LLM-ready output
`memory_search(query, limit?)`	Raw FTS keyword search, returns matching bullets
`memory_show(slug)`	Show all bullets for a node (read-only — does not clear budget)
`memory_add_bullet(node_slug, text, bullet_type?)`	Add a bullet to a node (auto-creates node if missing)
`memory_mark_reviewed(slug)`	Clear a node's token budget (marks node as reviewed)

Table 8: MCP tools exposed by `kg serve`

.claude → .kg Symlink

`kg start` creates `<project>/.claude → .kg`. This lets Claude Code automatically discover:

- The MCP server registration (`settings.json`)
- Local skills under `.kg/skills/` (accessible as `.claude/skills/`)
- Project-specific instructions in `.kg/CLAUDE.md` (if present)

> 🔥 **Bundled Skills**
>
> The `/note` and `/notes` skills ship with kg and are installed by `kg start`. `/note` adds a timestamped bullet with async cross-reference enrichment. `/notes` lists recent daily notes.

Session Context Hook

`kg start` installs a `UserPromptSubmit` hook that injects the current session ID into each prompt:

```json
{
  "hooks": {
    "UserPromptSubmit": [{
      "hooks": [{ "type": "command", "command": "python -m
      kg.hooks.session_context" }]
    }]
  }
}
```

This enables per-session deduplication: bullets already surfaced in the current conversation are filtered from future `memory_context` results.

Stop Hook: Fleeting Notes Extraction

The stop hook runs at session end and promotes fleeting notes to permanent nodes:

1. Reads `_fleeting-<session-id>` node
2. Identifies patterns worth promoting to permanent nodes
3. Adds bullets to appropriate nodes with `[[cross-references]]`
4. Deletes the fleeting bullets after promotion

```bash
# During a session — low-friction scratch notes:
kg add _fleeting-abc123 "discovered: lock file blocks concurrent kg reindex"

# At session end, the stop hook promotes this to the right permanent node.
```

Web Viewer

Overview

`kg web` serves a read-only UI at `http://localhost:7343`.

```bash
kg web              # start web server (default port 7343)
kg web --port 8080  # custom port
```

Routes

Route	Content
`/`	Node index: all curated nodes + collapsible docs section
`/node/<slug>`	Node page: bullets, backlinks, related nodes (lazy-loaded)
`/node/_doc-xxx`	Source file page: metadata, chunks, syntax highlighting, GitHub link
`/search?q=<query>`	Search results page (FTS + vector blend)
`/api/related/<slug>`	JSON endpoint for lazy-loading related nodes

Table 9: Web viewer routes

Features

- **Backlinks**: every node page shows all nodes that reference `[[this-slug]]` in their bullets
- **Related nodes**: lazy-loaded sidebar using the node title + first 6 bullets as query
- **Source file rendering**: syntax highlighting (highlight.js CDN) and Markdown rendering (marked.js CDN)
- **Auto-linkification**: file paths like `src/kg/web.py` in bullet text are auto-linked to their `_doc-*` nodes
- **GitHub link**: doc pages show a "GitHub ↗" link pointing to the file at its last-modified commit hash

Troubleshooting

SQLite Disk I/O Error

On virtualized filesystems (OrbStack, NFS, Docker volumes), `graph.db` can become a 0-byte empty file, causing `disk I/O error` on all `kg` commands:

```bash
# Fix:
kg stop
rm .kg/index/graph.db*
kg reindex        # stops watcher, rebuilds, restarts
```

> **⚠ Warning**
>
> Always stop the watcher **before** removing `graph.db`. The watcher holds WAL/SHM files; deleting them while the watcher runs causes corruption on the next write.

Zero Files Indexed from Sources

If `kg status` shows 0 files indexed for a source:

- Check include patterns: use `**/*.py` not `*.py` (see Python 3.12 fnmatch note)
- If `use_git = true`: verify with `git ls-files` in the source directory that files are tracked
- Run `kg index` to trigger an immediate re-index cycle
- Check `kg status` config section for path/git validation errors

Embeddings Coverage Below 100%

After adding many nodes while the vector server was stopped, some nodes may have no embeddings. Start the server and recalibrate:

```bash
kg start      # ensure vector server is running
kg calibrate  # regenerates missing embeddings + recalibrates scores
```

Vec Scores Missing from Calibration

If `kg status` shows "no vec calibration", the vector server was likely stopped during the last `kg calibrate` run:

```bash
kg start      # start the vector server
kg calibrate  # re-run calibration
```

Glossary

token budget: A credit counter per node tracking how many characters the node has contributed to context output. Triggers a review flag when credits-per-bullet exceeds the threshold. 12

bullet: A single atomic fact, decision, gotcha, or task belonging to a node. Stored as a JSONL line with a stable UUID. 1

calibration: The process of computing score quantile breakpoints from a random sample of nodes, used to normalize raw FTS and vector scores before fusion. 9

MCP – Model Context Protocol: Open protocol for exposing tools to LLMs. kg exposes memory_context, memory_search, memory_show, and memory_add_bullet. 13

node: A named collection of bullets identified by a slug. Stored as a directory under `.kg/nodes/<slug>/`. 1

reranker: A cross-encoder model (ms-marco-MiniLM-L-6-v2) that rescores the top-N hybrid search results using the full query-bullet pair. 9

slug: A URL-safe lowercase identifier for a node, e.g. `my-topic`. Pattern: `[a-z0-9][a-z0-9-]*`. 1

watcher: A background process that uses inotify (Linux) to detect changes to `.kg/nodes/` and immediately re-indexes modified nodes into SQLite. 2