Quarry

Your LLM can finally search your hard drive

February 2026 | JM Freeman | Open Source

Press Release

Summary

Today, the Quarry project announced the general availability of Quarry, a free, open-source tool that gives Claude instant access to everything on your computer. Quarry indexes PDFs, scanned documents, spreadsheets, presentations, web pages, notes, source code, and images into a local semantic search engine, then exposes that knowledge to Claude Code, Claude Desktop, and Cowork through the Model Context Protocol. Unlike uploading files to Projects or pointing your LLM at folders, Quarry lets Claude search your entire knowledge base by meaning—across every format, with no manual organization required.

Problem

People who use Claude for serious work—refactoring codebases, analyzing research, drafting from reference materials—hit the same wall every session. They spend 5–15 minutes gathering context before they can ask their real question. They upload PDFs to Projects. They copy-paste from old documents. They point Claude Code at directories and hope the right files get picked up. When the context window fills, they start over.

The frustration compounds. A developer with ten years of reference PDFs, design documents, and archived codebases cannot make any of that knowledge available to Claude without manually curating it first. A researcher with hundreds of papers must decide which three are relevant before the conversation starts. The LLM is powerful, but it can only reason about what it can see—and most of what you know is invisible to it, trapped in files you haven't opened in months.

Solution

Quarry works in three steps: install, ingest, search. Run `pip install quarry-mcp` and `quarry install`. Point it at your files—`quarry ingest report.pdf`, or register an entire directory with `quarry register ~/Documents/research`. From that moment, Claude can search your indexed knowledge semantically, finding relevant passages by meaning rather than keywords.

Every format is handled automatically. Text PDFs are extracted. Scanned documents and images are OCR'd—locally, with no cloud account. Spreadsheets are serialized to LaTeX tabular format. Presentations are split slide-by-slide. HTML pages are stripped of boilerplate and converted to Markdown. Source code is parsed into semantic units using Tree-sitter. Markdown, LaTeX, and DOCX files are split by section structure. The result is the same: chunks of knowledge that Claude can retrieve and reason about, with metadata indicating the document, page, content type, and source format.

Because Quarry runs entirely on your machine, your documents never leave your laptop. There is no cloud service, no API key to manage, no subscription. The vector database, the embedding model, and the OCR engine all run locally.

Customer Quote

> *"I have eight years of architecture decision records, design docs, and reference papers on this laptop. Before Quarry, Claude couldn't see any of it. I'd spend the first ten minutes of every session uploading the three PDFs I thought were relevant—and half the time I picked the wrong ones. Now I just ask my question and Claude finds what it needs. Last week it pulled a constraint from a 2019 design doc I'd completely forgotten about. That one save paid for the twenty minutes it took to set up."*
> — **Sarah Lindqvist**, Staff Engineer at a Series C startup *(composite persona representing the target user)*

Getting Started

Getting started takes under five minutes and requires no accounts, API keys, or configuration:

1. **Install:** `pip install quarry-mcp && quarry install` downloads the embedding model and configures Claude Code and Claude Desktop automatically.
2. **Ingest:** `quarry register ~/Documents/work && quarry sync` indexes an entire directory. Or ingest individual files with `quarry ingest report.pdf`.
3. **Search:** Open Claude Code or Claude Desktop and ask a question. Claude calls Quarry's search tool automatically—your indexed documents are now part of every conversation.

There is no cloud account to create, no API key to configure, and no subscription to manage. Everything runs on your machine.

Spokesperson Quote

> "The bottleneck for LLM-assisted work isn't the model—it's the context. People have years of accumulated knowledge on their hard drives, and none of it is accessible to their LLM. We built Quarry because we believe the knowledge you already have should be available in every conversation, without you having to remember where it is or manually upload it. The Model Context Protocol made this possible: Quarry doesn't just index your files, it makes them a native part of how Claude thinks."
>
> — **JM Freeman**, Creator of Quarry

Call to Action

Quarry is available now at `https://pypi.org/project/quarry-mcp/` and `https://github.com/jmf-pobox/quarry-mcp`. It is free, open source (MIT license), and always will be. Install it with `pip install quarry-mcp`, run `quarry install`, and start searching.

Frequently Asked Questions

External FAQs

Q: What is Quarry and who is it for?

Quarry is a local semantic search engine for your files that integrates with Claude through MCP. It is for people who use Claude Code, Claude Desktop, or Cowork for real work—writing, coding, research, analysis—and are tired of manually uploading files or organizing context folders before every session. If you have documents on your hard drive that you wish Claude could see, Quarry is for you.

Q: How is this different from uploading files to Claude Projects?

Projects require you to select and upload files before every conversation. You choose which documents are relevant, the context window limits how many you can include, and when you start a new conversation, you start from scratch. Quarry inverts this: you index your files once, and Claude searches them on demand during any conversation. You don't decide what's relevant—Claude does, by searching semantically across everything you've indexed. There is no context window limit because Quarry returns only the relevant passages, not entire files.

Q: How is this different from pointing Claude Code at a folder?

When Claude Code reads a directory, it sees file contents but has no semantic index. It reads files sequentially, consuming context window with every file it opens. For a folder with 50 documents, Claude Code might exhaust its context before finding the relevant passage. Quarry pre-indexes the content and returns only the matching chunks, leaving the context window available for reasoning rather than reading.

Q: What file formats does Quarry support?

PDF (both text and scanned/image pages), images (PNG, JPG, TIFF, BMP, WebP), spreadsheets (XLSX, CSV), presentations (PPTX), HTML, text files (Markdown, LaTeX, plain text, DOCX), and source code in 30+ languages. Every format is converted to text optimized for LLM consumption. Spreadsheets and presentation tables are serialized to LaTeX to preserve tabular structure. Scanned documents and images are OCR'd locally—no cloud account needed.

Q: Does my data leave my computer?

No. Quarry runs entirely on your machine. The vector database (LanceDB), the embedding model (snowflake-arctic-embed-m-v1.5 via ONNX Runtime), and the OCR engine (RapidOCR) all run locally. No data is sent to any cloud service. The only optional cloud component is AWS Textract for higher-quality OCR of degraded scans, and that is opt-in.

Q: How much does it cost?

Free. Quarry is open source under the MIT license. There is no paid tier, no usage limits, and no telemetry. The embedding model download is approximately 500 MB; after that, everything runs offline.

Q: Can I use Quarry with tools other than Claude?

Quarry exposes a standard MCP server over stdio. Any MCP-compatible client can use it. Today that includes Claude Code, Claude Desktop, and Cowork. As other LLM tools adopt MCP, Quarry will work with them without modification. The CLI also works standalone for direct search without any LLM.

Q: How does search quality compare to keyword search (Spotlight, grep)?

Quarry uses semantic search: it finds passages by meaning, not by matching words. Searching for "authentication logic" will find a function called `verify_credentials` even though the words don't overlap. This is the same retrieval approach used by commercial RAG systems, running locally with a 768-dimension embedding model.

Internal FAQs

Value & Market

Q: What is the total addressable market?

Bottoms-up: Claude Code had 115,000 developers by July 2025 and doubled its weekly active users between January and February 2026, with annualized revenue exceeding $2.5B. Claude Desktop and Cowork add to this base. Among these users, the power-user segment—those who have moved beyond simple Q&A to using Claude for complex, multi-document work—is the target. Conservatively estimating this at 10–20% of Claude tool users gives a TAM of hundreds of thousands of individuals who would benefit from local document search. Because Quarry is free, TAM is measured in adoption rather than revenue. Success means becoming the default way Claude users make their local files accessible.

Q: What evidence do we have that users want this?

Three categories of evidence:

Behavioral: Claude Desktop's Projects feature exists specifically because users want to give Claude access to their documents. The friction of uploading files and the context window limits are well-documented pain points in the Claude community. Users routinely ask in forums how to make Claude "see" their local files.

Competitive: Multiple personal-knowledge startups (Rewind/Limitless, Recall, Mem) raised significant funding on the premise that personal knowledge should be searchable by AI. Rewind rebranded to Limitless, was acquired by Meta in December 2025, and shut down its consumer product—validating the demand while demonstrating the fragility of cloud-dependent solutions. Quarry's local-first architecture is resilient to this exact failure mode.

Adjacent: Obsidian's plugin ecosystem shows that knowledge workers will invest significant effort to make their notes searchable by AI. Quarry eliminates the need for a specific note-taking app by working with any file format.

Q: Who are the competitors and why will we win?

Claude Projects / file uploads: The incumbent workflow. Quarry wins by eliminating the manual step entirely—index once, search forever, across all conversations.

Claude Code folder access: Reads files but lacks semantic indexing. Quarry wins on efficiency: returns relevant chunks instead of consuming context window reading entire files.

macOS Spotlight: System-level keyword search. Quarry wins on semantic matching and LLM integration—Spotlight can't find "authentication logic" in a function called `verify_credentials`.

RAG frameworks (LangChain, LlamaIndex): Powerful but require pipeline assembly, configuration, and Python expertise. Quarry wins on simplicity: three commands, zero configuration, works out of the box.

Cloud knowledge products (Mem, Notion AI): Require accounts, subscriptions, and sending data to third-party servers. The cautionary example is Rewind/Limitless: acquired by Meta in December 2025, consumer product shut down, users stranded. Quarry wins on privacy, cost (free), durability (local-first, open source), and MCP-native integration.

Structural advantage: Quarry's MCP integration means it is a native part of how Claude works, not an external tool that requires explicit invocation. Competitors would need to adopt MCP to match this, and by then Quarry will be the established default.

Q: What is the user acquisition strategy?

Three channels:

PyPI discoverability: Users searching for "claude mcp" or "local RAG" find Quarry on PyPI.

Claude community: Documentation, forum posts, and word-of-mouth among Claude Code and Desktop users. The "quarry install" command auto-configures MCP, reducing the setup barrier to near zero.

GitHub organic: The repository demonstrates engineering quality (comprehensive tests, strict typing, clean architecture) that attracts contributors and stars, improving discoverability.

A macOS menu bar companion app (`https://github.com/jmf-pobox/quarry-menubar`, in development) will surface Quarry to non-CLI users, expanding reach beyond the developer audience. The HTTP API server (`quarry serve`) is already shipped to support it.

Technical

Q: What are the major technical risks?

Embedding model quality (medium risk): The snowflake-arctic-embed-m-v1.5 model (768 dimensions, 512 token context) performs well for general text but may underperform on domain-specific content (medical, legal, financial jargon). Mitigation: the backend abstraction allows swapping models without code changes. A future model upgrade is a configuration change, not an architecture change.

OCR accuracy on degraded scans (low risk): The local RapidOCR backend produces good-enough text for semantic search but lower character accuracy than cloud OCR. Mitigation: Textract backend available for users who need higher fidelity. Semantic search is tolerant of OCR errors because it matches by meaning, not exact characters.

LanceDB scalability (low risk): LanceDB is designed for local, single-user workloads. At tens of thousands of documents, query latency and index size may become concerns. Mitigation: the database abstraction allows migration to a different backend if needed, and named databases allow partitioning by project.

Q: What dependencies exist on external systems?

MCP protocol: Quarry depends on the MCP specification remaining stable and on Claude clients continuing to support MCP tool servers. Risk is low: Anthropic is investing heavily in MCP as a standard, and multiple third-party clients now support it.

Embedding model hosting: The ONNX model is downloaded from Hugging Face Hub at install time. If the model is removed from the Hub, existing installations continue to work (model is cached locally), but new installations would need a mirror. Mitigation: model is pinned to a specific git revision for reproducibility.

Tree-sitter grammars: Source code parsing depends on Tree-sitter grammar packages. These are mature, widely-used open-source projects with low risk of abandonment.

No runtime cloud dependencies exist for the default configuration. Quarry operates fully offline after installation.

Q: What is the development timeline and current status?

Quarry is live on PyPI. Current capabilities:

Shipped: PDF ingestion (text + OCR), image OCR, spreadsheet ingestion (XLSX/CSV via LaTeX serialization), presentation ingestion (PPTX), HTML parsing, text file parsing, source code parsing (30+ languages via Tree-sitter), semantic search with metadata filtering (page type, source format), directory sync, named databases, MCP server, CLI, HTTP API server.

In progress: macOS menu bar companion app (`https://github.com/jmf-pobox/quarry-menubar`).

Future: Google Drive connector, incremental re-indexing on file change (`quarry sync -watch`), PII detection and redaction.

Q: What is the scaling story?

Quarry is designed for personal-scale workloads: thousands to tens of thousands of documents on a single machine. LanceDB handles this efficiently with on-disk storage and approximate nearest neighbor search.

Initial indexing is the main time investment. Benchmarked on an M2 MacBook Air, the ONNX embedding model sustains ~11 chunks/second (7–13 depending on batch size; see `benchmarks/embed_throughput.py`). Concurrent workers provide no additional throughput due to CPU cache contention. A 10,000-file directory (~70,000 chunks) takes roughly 1.5–2 hours. Files are queryable as they are indexed—the user does not wait for the full sync to complete. Subsequent syncs are incremental and fast (only new or changed files).

Query latency remains fast regardless of database size because LanceDB uses IVF-PQ indexing.

Quarry deliberately does not target multi-user or server deployments. The architecture assumes a single user on a single machine, which simplifies everything from concurrency to data privacy.

Business

Q: What is the revenue model?

There is none. Quarry is free, open-source software under the MIT license. The project exists to solve a real problem for the Claude user community and to demonstrate that local, privacy-preserving AI tooling can be simple and high-quality. Value is measured in adoption and community contribution, not revenue.

Q: What are the key metrics for success?

- **PyPI installs:** Monthly download count as a proxy for adoption. Current baseline: ~765 monthly downloads (February 2026). Target: 1,000 monthly installs within 6 months of launch.
- **GitHub stars:** Community interest signal. Target: 500 stars within 6 months.
- **Active MCP connections:** Users who have Quarry configured as an MCP server in Claude Code or Desktop (not directly measurable without telemetry, but inferable from support requests and community engagement).
- **Contributor count:** External contributors submitting PRs. Target: 5+ contributors within the first year.
- **Format coverage:** PDF, images, text, code, XLSX, PPTX, HTML are all shipped. Remaining targets: EPUB, email (EML/MBOX), audio transcription.

Q: Why now? What has changed?

Three converging shifts:

MCP maturity: The Model Context Protocol shipped in late 2024 and is now supported by Claude Code, Claude Desktop, and Cowork. For the first time, a local tool can be a native part of an LLM's reasoning process, not just an external API the user invokes manually.

Local inference viability: ONNX Runtime, quantized embedding models, and efficient vector databases (LanceDB) have made it feasible to run a complete semantic search pipeline on a laptop CPU without GPU, in under a second per query.

LLM power-user emergence: A critical mass of users have moved beyond ChatGPT-style Q&A to using LLMs for complex, multi-document tasks. These users are actively looking for ways to feed more context to their LLM and are frustrated by the manual overhead of doing so.

Q: What are we not building?

- **A cloud service.** Quarry will never require an account, subscription, or data upload. Local-first is a permanent design constraint, not a temporary limitation.
- **A multi-user platform.** No authentication, no access control, no shared databases. Quarry is a single-user tool for a single machine.
- **Media search.** Quarry does not find images by visual content or match audio. It extracts *text* from every format and indexes the knowledge inside.
- **A note-taking app.** Quarry indexes existing files. It does not provide an editor, a tagging system, or a filing structure.
- **A general-purpose RAG framework.** No pipeline assembly, no pluggable retrievers, no chain-of-thought orchestration. Quarry is deliberately opinionated and zero-configuration.

Q: What does failure look like?

Two plausible failure scenarios:

Obscurity: Quarry ships, works well, and nobody discovers it. After 6 months, monthly PyPI installs plateau below 200. The project becomes a personal tool rather than a community standard. *Response:* This is the most likely failure mode. Mitigation is the menu bar app (widens audience beyond CLI), active presence in Claude community forums, and ensuring `quarry install` auto-configures MCP so setup friction approaches zero. If installs plateau, invest in content marketing (blog posts, demo videos) before building more features.

First-party displacement: Anthropic ships native file indexing in Claude Code or Desktop, making third-party tools unnecessary. *Response:* If Anthropic builds this, it validates the thesis and serves users better than any third-party tool could. Quarry would still serve users who want

local-only indexing with no cloud dependency, offline access, or support for formats Anthropic doesn't cover. The open-source codebase remains useful as a reference implementation. This is a "good problem to have" failure.

Risk Assessment

Q: Value — Will users adopt this?

Medium risk. The demand is proven: Claude Projects and file uploads exist because users asked for a way to use their local files with AI. Forums and communities are full of people asking "is there an easier way to give Claude access to my documents?" Quarry is a direct answer to that question.

The risk is not demand—it is *discovery and activation*. Nobody will search for "MCP-based local semantic search." They will search for "how to use my files with Claude without uploading them." Quarry must be present where those questions are asked—Reddit, Claude community forums, Stack Overflow—with a one-sentence answer: "Stop uploading files. Install Quarry, index once, search forever."

Activation risk is also real. A user who indexes one file will not see the value. The "wow" moment—Claude retrieving a document the user forgot they had—requires enough indexed content to make semantic search surprising and useful. The onboarding must encourage bulk ingestion (directory registration) over single-file ingest.

The macOS menu bar app is the highest-leverage item on the roadmap for reducing both risks: it widens the audience beyond CLI users and makes Quarry's retrieval visible, creating shareable moments that drive word-of-mouth.

Q: Usability — Can users figure out how to use it?

Medium risk. For CLI-native users, three commands to value: `pip install`, `quarry install`, `quarry ingest`. Auto-configures MCP for Claude Code and Desktop. No accounts, no API keys, no configuration files. Once set up, the user does nothing—Claude invokes Quarry automatically during conversations.

However, the product currently requires Python, pip, and comfort with a terminal. For the non-CLI audience identified as a growth target in the Value section, usability risk is real. The macOS menu bar app (in development) is the primary mitigation: once shipped, it reduces the setup path to a native macOS install with no terminal required. Until then, Quarry is accessible only to developers and technical power users.

Q: Feasibility — Can we build it?

Low risk. The product is already built and live on PyPI. Core pipeline (ingest, embed, search, MCP server) is shipped across 10 document formats with 497 passing tests and strict type checking. The remaining roadmap items (menu bar app, Google Drive connector, `sync --watch`) are additive features, not architectural risks.

Q: Viability — Does it work as a project?

Low risk. No revenue model to validate. Infrastructure cost is zero (runs on the user's machine). Development cost is the maintainer's time. MIT license eliminates legal risk. The

project is compatible with Anthropic's ecosystem strategy and creates value for the Claude user community.