Free Plugin Gives Claude Code Persistent Memory Across Sessions

Quarry learns from every conversation and recalls what you've researched before—locally, with no cloud accounts or data leaving your machine

Open source plugin captures web research, session transcripts, and design decisions as searchable knowledge that compounds over time

Press Release

PORTLAND, OR — September 2026 — Today, Punt Labs announced Quarry Ambient Knowledge, a free Claude Code plugin that gives AI coding sessions persistent memory. Every web page Claude researches, every debugging conclusion reached, every design decision made is automatically captured into a local, searchable knowledge base. When Claude encounters a familiar topic in a future session, it checks the knowledge base before searching the web—eliminating redundant research and preserving the reasoning behind past decisions. Unlike hosted context platforms that require cloud accounts and team infrastructure, Quarry runs entirely on the developer's machine, is open source under the MIT license, and works with the Claude Code plugin system that has grown to over 9,000 extensions [1] (see FAQ 9).

Problem

Developers who use Claude Code for serious, multi-session work—refactoring systems, debugging production issues, designing architectures—lose everything they learn at the end of every conversation. A developer spends forty-five minutes with Claude tracking down a NATS connection timeout, reads three Stack Overflow answers, identifies the root cause, and ships the fix. Two weeks later, the same class of issue appears in a different service. Claude starts from zero—searches the web again, reads the same Stack Overflow answers, discovers the same root cause. The developer recognizes the pattern but Claude does not, because Claude has no memory of the previous session.

The problem extends beyond debugging. A developer writes a PR/FAQ using Claude, generating competitive analysis, customer evidence, and risk assessments through hours of structured conversation. The next PR/FAQ starts from scratch—none of the prior research is available. A team lead reviews a Z specification, debating invariants and edge cases for an afternoon. That reasoning exists nowhere except in a conversation transcript that was compressed and discarded.

The knowledge is generated. The work is done. And then it vanishes (see FAQ 10).

Solution

Quarry Ambient Knowledge installs as a Claude Code plugin and operates through two loops that run without user intervention.

The learning loop captures knowledge passively. When Claude fetches a web page, a hook queues the URL for background ingestion. Before context compaction, a hook captures the conversation transcript. When Claude reads a PDF or spreadsheet, it is queued for indexing. The developer works normally; the knowledge base grows from the natural flow of every session. Capture selectivity is controlled by a single command—`/quarry learn on`—and all hooks are fail-open: if Quarry crashes, Claude Code continues uninterrupted (see FAQ 3).

The recall loop surfaces relevant knowledge at the right moment. At session start, Claude receives a briefing: "You have 347 documents across 5 collections. Use `/find` to search locally before web searching." Before Claude initiates a web search, a hook checks whether the query overlaps with locally indexed content. If it does, Claude sees a nudge: "Quarry has 8 documents about LanceDB indexed locally. Consider checking locally first." Claude can still search the web—the hook does not block—but it knows to check its own memory first (see FAQ 4).

Customer Quote

> *"I've been using Claude Code full-time for seven months. The thing that frustrated me most wasn't the model—it was the amnesia. I'd debug something, figure out the fix, and three weeks later Claude would research the same problem from scratch while I watched. With Quarry's learning mode on, the web pages from that first session were already indexed. The second time, Claude found the answer locally in two seconds instead of spending five minutes searching Stack Overflow. After a month, I had a searchable archive of every piece of research Claude had ever done for me—and I never explicitly saved any of it."*
>
> — **Sarah Lindqvist**, Staff Engineer at a Series C startup

Getting Started

Getting started requires no accounts, no API keys, and takes under three minutes:

1. **Install the plugin:** Run `claude plugin install quarry@punt-labs` to configure Quarry with hooks, commands, and MCP tools. The embedding model downloads automatically on first use.
2. **Enable learning:** Type `/quarry learn on` in any Claude Code session. From this moment, web research and compaction transcripts are captured automatically. Type `/quarry learn all` to also capture document reads and agent findings.
3. **Work normally.** There is no third step. The knowledge base builds itself as you work. Search it anytime with `/find "your question"`.

The recall loop is always active once the plugin is installed—Claude knows about your knowledge base from the first session. Learning can be enabled or disabled per-project at any time.

Spokesperson Quote

> "We built the original Quarry to unlock knowledge trapped in files on your hard drive. But we realized the bigger problem isn't static files—it's the knowledge generated during every coding session that vanishes when the conversation ends. Web research, debugging conclusions, design decisions—all of it evaporates. Ambient Knowledge turns Quarry from a tool you call into a layer that makes Claude Code itself smarter. The host changes behavior because the plugin is present. That's not a search engine—that's memory."
>
> — **JM Freeman**, Creator of Quarry

Call to Action

Quarry Ambient Knowledge will be available as a free Claude Code plugin at `https://github.com/punt-labs/quarry`. The plugin, the CLI, and the underlying search engine are all open source under the MIT license. Install with `claude plugin install quarry@punt-labs`, enable learning, and let your knowledge compound.

Frequently Asked Questions

External FAQs

Q1: What is Quarry Ambient Knowledge and who is it for?

Quarry Ambient Knowledge is a Claude Code plugin that gives your coding sessions persistent memory. It passively captures web research, conversation transcripts, and document reads into a local semantic search engine, then surfaces that knowledge in future sessions before you search the web again. It is for developers who use Claude Code for multi-session work and are tired of Claude re-researching topics it has already investigated.

Q2: How is this different from CLAUDE.md or memory files?

CLAUDE.md is static instructions that you write and maintain manually. It tells Claude *how* to behave. Quarry captures *what Claude learned*—web pages, debugging conclusions, design rationale—and makes it searchable by meaning. You do not write or maintain the knowledge base; it builds itself from every session. CLAUDE.md says "use ruff for linting." Quarry remembers that the last time you debugged a ruff configuration issue, the root cause was a conflicting pyproject.toml section, and the Stack Overflow answer that explained it.

Q3: How does the learning loop actually work?

Quarry uses Claude Code's hook system—the same event-driven architecture used by plugins like Entire.io for code provenance. When Claude fetches a web page, a `PostToolUse` hook writes the URL to a staging queue. Before context compaction, a `PreCompact` hook captures the conversation transcript. All hooks write to a local staging directory and return immediately—they never block the conversation. A background `quarry learn` process ingests the queue: fetching pages, chunking text, generating embeddings, and indexing into LanceDB. The process is fail-open: if Quarry is unavailable, Claude Code works exactly as before.

Q4: How does the recall loop work?

Two hooks make Claude aware of your knowledge base. At session start, a `SessionStart` hook injects a brief summary into Claude's context: how many documents are indexed, which collections exist, when the last sync occurred. Before Claude initiates a web search, a `PreToolUse` hook checks whether the search query overlaps with indexed topics. If it does, Claude receives a nudge suggesting a local search first. The hook does not block web search—Claude decides whether local knowledge is sufficient or whether fresh web results are needed.

Q5: 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 all hooks execute locally. No data is sent to any cloud service. Captured web pages are re-fetched and stored locally, not forwarded through a proxy.

Q6: 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.

Q7: Can I control what gets captured?

Yes. Three modes via `/quarry learn`:

- **`off`** — No passive capture. Quarry works as a manual search tool only.
- **`on`** — Captures web research (URLs Claude fetches) and conversation transcripts (before compaction). High signal, low volume.
- **`all`** — Also captures non-code document reads (PDFs, spreadsheets, images), research agent findings, and session digests. Higher volume, broader coverage.

Each mode is a superset of the previous. Learning can be changed per-project at any time.

Q8: Does Quarry also work as a standalone search tool?

Yes. Quarry was originally built as a local semantic search engine for files on your hard drive—PDFs, scanned documents, spreadsheets, presentations, source code, web pages, and 30+ other formats. The ambient knowledge plugin layer is additive. Everything the original Quarry does—manual ingestion, directory sync, named databases, CLI search, HTTP API, macOS menu bar app—continues to work. The plugin adds passive learning and active recall on top.

Internal FAQs

Value & Market

Q9: What is the total addressable market?

Claude Code had over 115,000 developers by mid-2025 [2] and has been growing rapidly, with Anthropic reporting doubled weekly active users and annualized revenue exceeding $2.5B by early 2026 [3]. The plugin ecosystem grew from zero to 9,000+ extensions in under five months [4]. The broader AI-assisted coding market includes users of Cursor, GitHub Copilot, and Windsurf, totaling millions of developers.

The target segment is *multi-session power users*—developers who use AI coding tools for complex work spanning days or weeks, not single-turn Q&A. The Qodo survey found that 41–52% of developers report context-related frustration [5]. If even a quarter of frustrated users are multi-session Claude Code users who would benefit from passive capture, the segment is tens of thousands. This derivation is an assumption—validating the actual multi-session usage rate is part of the Phase 1 recall hook experiment. Because Quarry is free, TAM is measured in adoption, not revenue. Success means becoming the default way Claude Code sessions persist knowledge.

Q10: What evidence do we have that developers want this?

Four categories of evidence:

Survey data: The Qodo "State of AI Code Quality" survey found that context-related frustration affects 41–52% of developers (increasing with seniority), and "improved contextual understanding" is the single most-requested AI improvement at 26% of votes. The survey measured three tiers of context management: manual selection (54% frustration), autonomous tool selection (33%), and persistent storage with reuse (16%) [5]. Quarry targets the third tier—persistent context that compounds across sessions. The Stack Overflow 2025 survey (49,000+ respondents) found 81% have security/privacy concerns about AI data use [6]—local-first architecture directly addresses this.

Market signal: Multiple startups are building products for this exact problem. SageOx [7], an early-stage startup (funding status undisclosed), positions itself as "context infrastructure for human-agent collaboration"—capturing agent sessions as durable team artifacts. Entire.io, founded by former GitHub CEO Thomas Dohmke with $60M in seed funding [8], captures the reasoning behind AI-generated code via a git shadow branch [9]. Both validate that knowledge loss between agent sessions is a problem worth solving.

Behavioral: Claude Code's own memory features (CLAUDE.md, auto-memory) exist because developers asked for session persistence. These features are static and manually maintained; they capture instructions, not learned knowledge. The gap between "remember my preferences" and "remember what we researched" is the gap Quarry fills.

Adjacent: The cautionary example is Rewind/Limitless—a personal knowledge capture tool that raised $33M+, was acquired by Meta in December 2025, and shut down both its Pendant hardware and Rewind desktop app [10, 11]. The demand was real; the cloud-dependent model was fragile. Quarry's local-first architecture is resilient to this failure mode.

Q11: Who are the competitors and why will we win?

SageOx: Hosted context platform for enterprise teams [7]. Captures agent sessions and team discussions as structured artifacts. Differentiators: team collaboration, a web UI for browsing shared history, and structured extraction of decisions and intent. Quarry is not competing for enterprise teams—it serves individuals and small teams with a local-first, open-source tool that requires no accounts or infrastructure.
Entire.io: Local-first session recording for individual developers [9]. Uses a git shadow branch to store session metadata alongside code. Differentiator: deep git integration and checkpoint/rewind functionality. Quarry's differentiator is semantic search—not just recording what happened, but making it searchable by meaning across sessions, formats, and topics.
Windsurf Cascade Memories: Built-in cross-conversation persistent context in the Windsurf AI code editor [12]. Remembers project details across sessions. However, it is cloud-hosted, Windsurf-specific, and does not support local-first data or cross-tool search. Quarry works with any Claude Code session and stores everything locally.
CLAUDE.md / auto-memory: Built-in Claude Code features for session persistence. These capture static instructions and preferences, not dynamic knowledge. They are complementary, not competitive—CLAUDE.md says how to work; Quarry remembers what was learned.
Structural advantage: Quarry is already a shipping tool with 30+ format support, a CLI, MCP server, HTTP API, and macOS menu bar app. The ambient knowledge layer builds on proven infrastructure rather than starting from scratch.
Platform risk: The most dangerous competitive response is Anthropic building ambient memory natively into Claude Code. If Claude Code ships a built-in session persistence feature, a plugin becomes redundant for the basic case. Quarry's residual value in that scenario: (1) cross-tool portability—the knowledge base works outside Claude Code (menu bar app, CLI, HTTP API, Claude Desktop); (2) user-owned data in an open format (LanceDB, not a proprietary store); (3) 30+ format indexing beyond what a native feature would likely support; (4) open source, auditable, and extensible. The honest answer: if Anthropic builds this natively, Quarry's ambient layer becomes a power-user extension rather than the default. That is an acceptable outcome for a free, open-source tool.

Q12: What is the user acquisition strategy?

Quarry already has an installed base from the original search tool (PyPI downloads, GitHub presence). The ambient knowledge features are delivered as a plugin update—existing users get the new capabilities automatically. New users discover Quarry through the same channels: PyPI, GitHub, Claude Code plugin marketplace, and Claude community forums.
The key activation metric is enabling learning. The plugin installs with learning off (backward compatible). A first-session prompt suggests `/quarry learn on` with a brief explanation. The value becomes obvious within a few sessions as the recall loop surfaces previously researched knowledge.

Q13: What is the next step to validate this vision?

Build the recall hooks (SessionStart briefing, PreToolUse WebSearch nudge) and deploy to the author's daily workflow for two weeks. These hooks require no new infrastructure—they read from the existing Quarry database and inject context via Claude Code's hook system. If the recall hooks demonstrably reduce redundant web searches in real usage, proceed to the learning hooks. If not, the premise is wrong and the ambient layer is not worth building.

This validation is cheap: the recall hooks are read-only, fail-open, and can be disabled in one command. The learning hooks (which write data) are deliberately sequenced after recall validation to avoid building capture infrastructure for a recall mechanism that does not work.

Technical

Q14: What are the major technical risks?

PreToolUse nudge precision (medium risk): The recall hook that nudges Claude before web searches needs to be fast (<50ms) and low-false-positive. A naive keyword match against collection names may produce too many false nudges, training the model to ignore them. Mitigation: start with a conservative matcher (exact collection name overlap), measure nudge acceptance rate, and iterate.

PreCompact data availability (medium risk): The learning hook that captures conversation transcripts depends on what data Claude Code exposes to PreCompact hooks. If the raw transcript is not available in hook stdin, the capture mechanism needs a different approach (e.g., session log file reading). Mitigation: test PreCompact hook data format before building the capture pipeline.

Staging queue reliability (low risk): Hooks write to a file-based staging queue. If the queue is corrupted or the ingestion process crashes mid-way, some captured knowledge may be lost. Mitigation: the queue is append-only text files (URLs, file paths); partial processing leaves the queue in a recoverable state. Lost captures are low-cost—the knowledge can be re-researched.

Hook latency (low risk): All hooks must return fast to avoid blocking Claude Code. The learning hooks write to files and return immediately—no embedding or network calls. The recall hooks read local state (database stats, collection names) which is cached and fast.

Q15: What dependencies exist on external systems?

Claude Code hook system: The entire ambient layer depends on Claude Code's hook events (PostToolUse, PreCompact, SessionStart, PreToolUse) remaining stable and available to plugins. Risk is low: the hook system is actively developed and used by multiple shipping plugins (Entire.io, tts, biff).

Existing Quarry infrastructure: The ambient layer builds on the shipping Quarry search engine—LanceDB for storage, ONNX embedding model for vectors, the CLI for ingestion. No new infrastructure is required. The learning hooks queue work for existing CLI commands; the recall hooks read from the existing database.

No new external dependencies are introduced.

Q16: What happens as the knowledge base grows?

LanceDB is designed for datasets far larger than a personal knowledge base. Search latency remains sub-second up to millions of vectors; a knowledge base of 10,000–50,000 documents (years of heavy use) is well within comfortable range. Disk usage scales at approximately 1–5 MB per 100 documents for embeddings and metadata. Raw captured content (web pages, transcripts) adds more but is bounded by the staging queue's dedup logic—the same URL is not re-ingested.

The PreToolUse nudge hook checks topic overlap via a lightweight keyword match against collection names and document titles, not a full embedding search. This stays under 50ms regardless of database size because it reads a cached index, not the full vector store.

If disk usage becomes a concern, the existing `quarry delete` command lets users prune old or low-value collections. A future "retention policy" feature (not in scope) could automate this.

Q17: What is the development timeline?

The implementation sequence is designed so each phase delivers standalone value:

1. **Recall hooks** (1–2 weeks): SessionStart briefing and PreToolUse WebSearch nudge. Immediate value with the existing knowledge base. Validates the core premise.
2. **Config layer** (1 week): `.quarry/config.md`, `/quarry learn` command, config reading in hooks.
3. **Learning hooks: web capture** (1–2 weeks): PostToolUse on WebFetch/WebSearch, staging queue, `quarry learn` CLI command.
4. **Learning hooks: compaction** (1 week): PreCompact transcript capture.
5. **Signal accumulation and extended capture** (1–2 weeks): `learn_signals` gating, document reads, agent findings.

Total: 5–8 weeks of incremental development, each phase shippable independently.

Business

Q18: What is the revenue model?

There is none. Quarry is free, open-source software under the MIT license. The ambient knowledge layer is a plugin feature, not a paid tier. Value is measured in adoption and ecosystem contribution. The project demonstrates that local-first, privacy-preserving AI tooling can be both simple and high-quality.

Q19: What are the key metrics for success?

- **Learning enablement rate:** Percentage of plugin users who activate `/quarry learn`. Target: 30% within 3 months of release. If users install the plugin but do not enable learning, the value proposition is unclear.
- **Recall hit rate:** Percentage of PreToolUse nudges where Claude uses the local knowledge instead of web searching. Target: >50% nudge acceptance. If Claude ignores the nudges, the recall mechanism needs refinement.
- **Knowledge base growth:** Average documents per user per month from passive capture. Expected: 50–200 web pages + 10–30 session notes per active month. If growth is lower, the capture hooks are not firing or the staging queue is failing.
- **PyPI downloads:** Monthly download growth as a proxy for overall adoption. Baseline: ~765/month (February 2026). Target: 2,000/month within 6 months of ambient knowledge release.

Q20: Why now? What has changed?

Two shifts converged:

Claude Code plugin ecosystem maturation. The hook system (PostToolUse, PreToolUse, PreCompact, SessionStart) is now stable and used by shipping plugins [13]. The ecosystem grew to 9,000+ plugins in under five months [4]. Entire.io's launch in February 2026 [8] demonstrated that the hook architecture supports deep session integration—capturing, gating, and injecting context at every stage of the agent lifecycle. The infrastructure for ambient behavior exists; Quarry applies it to knowledge persistence.

AI coding sessions became the primary research context. Developers increasingly do their research *inside* Claude Code—web searching, reading documentation, analyzing codebases—rather than in a browser. The knowledge generated during these sessions is higher-value than general browsing because it is already contextualized to the developer's work. But it vanishes when the session ends. The tool that captures and recalls this knowledge does not exist yet.

Q21: What is the opportunity cost?

The ambient knowledge layer requires 5–8 weeks of incremental development. Alternatives for the same time investment include: improving core search quality (relevance tuning, hybrid search), expanding the menu bar app (the highest-leverage adoption surface), or adding new format support.

The argument for ambient knowledge over these alternatives: the other improvements are incremental refinements to an existing tool. Ambient knowledge changes the category—from "a tool you call" to "a layer that makes the host smarter." The incremental improvements grow the existing user base linearly; ambient knowledge creates a new reason to install Quarry that did not exist before.

Ongoing maintenance cost: hook system changes in Claude Code require adaptation (estimated 1–2 days per breaking change, historically infrequent). User-side cost: the embedding model is a one-time 500 MB download; knowledge base disk growth is approximately 1–5 MB per 100 captured documents (embeddings + metadata, not raw content).

Q22: What are we not building?

- **A hosted platform.** No accounts, no cloud infrastructure, no web UI. Quarry is local-first permanently, not temporarily.
- **Multi-user collaboration infrastructure.** No shared databases, no access control, no team dashboards. Small-team sharing via a git shadow branch is a potential future direction, but not a launch requirement.
- **Structured knowledge extraction.** SageOx extracts summaries, decisions, and intent from captured content. Quarry indexes raw content and lets semantic search surface what is relevant. Raw indexing preserves everything; structured extraction is lossy.
- **Code provenance tracking.** Entire.io links commits to the sessions that produced them via checkpoint trailers. Quarry captures session transcripts that naturally contain this reasoning, but does not add metadata to git commits. Provenance is a byproduct of captured knowledge, not a primary feature.
- **An AI agent framework.** Quarry does not orchestrate agents, manage tool chains, or provide retrieval-augmented generation pipelines. It is a knowledge layer that other tools can query.

Q23: What does failure look like?

Two plausible failure scenarios:

Recall is ignored. Claude Code receives the session briefing and PreToolUse nudges but consistently web-searches anyway. The recall loop adds latency without changing behavior. *Response:* This is testable in two weeks with the recall hooks alone (Phase 1). If nudge acceptance rate is below 20% after tuning, the recall mechanism is not viable in its current form. Pivot to explicit recall (`/find` only) and focus on making the learning loop valuable for manual search.

Capture noise exceeds signal. The knowledge base fills with low-value web pages and irrelevant transcript fragments, degrading search quality. When the user does search, results are polluted by auto-captured noise. *Response:* This is why `/quarry learn on` starts conservative (web

research + compaction only). If signal-to-noise is poor even at the `on` level, add quality filtering—URL pattern exclusions, minimum page length, dedup by content similarity. The `all` mode exists precisely to gate the higher-noise sources behind an explicit user choice.

Risk Assessment

Risk	Rating	Assessment
Value	Medium	Multiple startups and survey data validate the problem [5, 8]. No direct evidence that *local, passive* capture changes developer behavior. Recall hook acceptance rate is the key unknown.
Usability	Medium	Install is frictionless. The recall nudge—the core UX—is a novel interaction pattern between hooks and the model with no reference class. Nudge acceptance rate is testable in Phase 1.
Feasibility	Low	All infrastructure exists: hooks, CLI, LanceDB, embedding model. Pre-Compact data format and nudge precision are medium-risk unknowns (see FAQ 14); overall feasibility is low because alternatives exist for each.
Viability	Low	No revenue model to validate. Zero marginal cost (runs on user hardware). MIT license.

Value risk is the dominant concern. The thesis is that passive capture plus automatic recall changes how developers work with AI coding tools—that the second time you research something should be instant. This is plausible but unvalidated. **Usability risk** is elevated because the recall nudge—the core value-delivery mechanism—is a novel interaction pattern between hooks and the model with no reference class; nudge acceptance rate is the key metric. The validation plan (see FAQ 13) tests recall hooks first because they are the mechanism that delivers value. If recall does not change behavior, the learning hooks are infrastructure without a product.

Feature Appendix

Defines the scope boundary for Quarry Ambient Knowledge. Features are categorized to prevent scope creep and force prioritization.

Must Do

Features essential for the ambient knowledge layer to deliver value.

- **F1.** **SessionStart knowledge briefing** — Claude must know the knowledge base exists before it can use it
- **F2.** **PreToolUse WebSearch nudge** — the recall mechanism that eliminates redundant web research
- **F3.** **Config layer** — per-project learning mode via config file, paralleling tts plugin pattern
- **F4.** **Learn command** — user-facing `/quarry learn` for capture selectivity: off, on, all
- **F5.** **PostToolUse WebFetch/WebSearch capture** — auto-queue URLs Claude researches for background ingestion
- **F6.** **Staging queue + `quarry learn` CLI** — file-based queue processed asynchronously, never blocking the conversation
- **F7.** **PreCompact transcript capture** — save conversation content before context compression

Should Do

Features that meaningfully improve ambient knowledge but are not launch-blocking.

- **F8.** **Signal accumulation (`learn_signals`)** — track what was learned per session to gate digest quality
- **F9.** **PostToolUse Read capture (non-code files)** — auto-index PDFs, spreadsheets, images Claude reads
- **F10.** **SubagentStop capture** — save research agent findings before they are compressed
- **F11.** **SessionEnd knowledge digest** — structured summary of what was learned, tagged with commit hashes
- **F12.** **Shadow branch storage** — captured content on a git branch for small-team sharing
- **F13.** **URL pattern exclusions** — filter out low-value web captures by URL pattern

Won't Do

Features explicitly excluded from this effort.

- **F14.** **Hosted web UI** — team browsing of shared knowledge—SageOx's territory, not ours
- **F15.** **Structured knowledge extraction** — extracting summaries, decisions, and intent from raw content adds complexity and is lossy—semantic search over raw content is sufficient
- **F16.** **Git commit metadata** — adding checkpoint trailers or session IDs to git commits—Entire.io's territory
- **F17.** **Multi-user access control** — authentication, permissions, shared databases—enterprise infrastructure we deliberately avoid
- **F18.** **Real-time sync between machines** — knowledge sharing via git shadow branch is async, not real-time

References

[1] AI Tool Analysis. *Claude Code Plugins Review 2026: 9,000+ Extensions*. Reports 9,000+ Claude Code plugins as of early 2026. 2026. URL: https://aitoolanalysis.com/claude-code-plugins/.

[2] ppc.land. *Claude Code reaches 115,000 developers, processes 195 million lines weekly*. Reports figures disclosed by Deedy Das (Menlo Ventures) on July 6 2025. 115,000 developers, 195M lines per week. Anthropic-confirmed July 2025 stat: 5.5x revenue increase. 2025. URL: https://ppc.land/claude-code-reaches-115-000-developers-processes-195-million-lines-weekly/.

[3] Anthropic. *Anthropic raises $30 billion in Series G funding at $380 billion post-money valuation*. Primary Anthropic source: Claude Code WAU doubled since January 1 2026; annualized revenue exceeded $2.5B; total Anthropic ARR $14B. Announced February 12 2026. 2026. URL: https://www.anthropic.com/news/anthropic-raises-30-billion-series-g-funding-380-billion-post-money-valuation.

[4] StartupHub AI. *Anthropic's Claude Code plugins open the floodgates*. Claude Code plugins launched public beta October 9 2025. Ecosystem grew to 9,000+ plugins in under five months. 2025. URL: https://www.startuphub.ai/ai-news/ai-research/2025/anthropics-claude-code-plugins-open-the-floodgates/.

[5] Qodo. *State of AI Code Quality in 2025*. Developer survey: context pain affects 41–52% of developers by seniority; "improved contextual understanding" is the top-requested AI improvement (26% of votes); persistent context reduces frustration from 54% to 16%. Qodo, 2025. URL: https://www.qodo.ai/reports/state-of-ai-code-quality/.

[6] Stack Overflow. *2025 Stack Overflow Developer Survey*. 49,000+ respondents from 177 countries. 84% use AI tools; 46% distrust accuracy (up from 31%); 66% cite "almost right but not quite" as top frustration; 81% have security/privacy concerns about AI data use. Stack Overflow, 2025. URL: https://survey.stackoverflow.co/2025/.

[7] SageOx. *Context Infrastructure for Human-Agent Collaboration*. 2026. URL: https://sageox.ai (visited on 02/28/2026).

[8] TechCrunch. *Former GitHub CEO raises record $60M dev tool seed round at $300M valuation*. Confirms Thomas Dohmke (GitHub CEO 2021–2025) founded Entire; $60M seed at $300M valuation led by Felicis; largest-ever developer tools seed round. 2026. URL: https://techcrunch.com/2026/02/10/former-github-ceo-raises-record-60m-dev-tool-seed-round-at-300m-valuation/.

[9] Entire. *entireio/cli: Entire is a new developer platform that hooks into your git workflow*. Open-source Checkpoints CLI. Captures prompts, transcripts, tool calls, token usage. Stores on entire/checkpoints/v1 git shadow branch. Supports Claude Code, Gemini CLI, OpenCode, Cursor. 2026. URL: https://github.com/entireio/cli.

[10] TechCrunch. *Meta acquires AI device startup Limitless*. Meta acquired Limitless (formerly Rewind) December 5 2025. Founded by Dan Siroker and Brett Bejcek. Raised $33M+ from Sam Altman, First Round, a16z, NEA. Team joins Meta Reality Labs. 2025. URL: https://techcrunch.com/2025/12/05/meta-acquires-ai-device-startup-limitless/.

[11] 9to5Mac. *Rewind Mac app shutting down following Meta acquisition*. Confirms Rewind Mac desktop app and Pendant hardware shutdown effective December 2025. 2025. URL: https://9to5mac.com/2025/12/05/rewind-limitless-meta-acquisition/.

[12] CodeAnt AI. *AI Code Editors Showdown 2025: Cursor vs Windsurf vs Copilot Explained*. Documents Windsurf Cascade Memories: persistent cross-conversation context in a major AI code editor. 2025. URL: https://www.codeant.ai/blogs/best-ai-code-editor-cursor-vs-windsurf-vs-copilot.

[13] Anthropic. *Claude Code Hooks Reference*. 2026. URL: https://code.claude.com/docs/en/hooks (visited on 02/24/2026).