The Long Tail: Semantic Archaeology of 99 Single-Install Claude Code Plugins

Every name is a compressed intention. Someone built each of these, published it, and exactly one person installed it. That person might be the author. The name is the only thing most people will ever see. What follows is a reading of each name across three registers: **Logical** (what it most likely does), **Lateral** (what adjacent meanings the words carry), and **Hallucinatory** (what it could mean if you let the associations run). Confidence is marked where relevant.

THE UNCLASSIFIABLES

These are the ones that resisted even category-level sorting. They're the most interesting.

silince-gutnebrg-builder

Register	Reading
Logical	WordPress Gutenberg block editor integration for Claude Code. The typos ("silince" for "silence", "gutnebrg" for "gutenberg") suggest a non-native English speaker typing fast, or someone who doesn't care about spelling because the code works. High confidence this is a Gutenberg block builder.
Lateral	"Silence" + "Gutenberg" + "Builder." Gutenberg's press was the original compression algorithm — oral tradition collapsed into fixed, reproducible text. The press *silenced* the variability of handwritten copies. A silence-builder constructs through reduction, not addition. Every printed page is a commitment to one version of a text at the expense of all others. Entropy reduction through mechanical reproduction.
Hallucinatory	The typos are the message. "Silince" isn't silence — it's si-LINCE, the lince being lynx in Romance languages. A sharp-eyed cat watching. "Gutnebrg" rearranges to contain "gut" and "nberg" (as in Nuremberg, Gutenberg's neighbouring city). A builder of gut-feeling architectures in the Nuremberg tradition — which is to say, a tribunal. Something that judges content blocks by instinct rather than rule. The misspellings are a steganographic test: does the reader see through the surface to the structure? Very you.

ida-reverse-engineer

Register	Reading
Logical	IDA Pro (Interactive DisAssembler by Hex-Rays) integration. Takes compiled binaries and reconstructs human-readable code. The research confirmed HexRaysSA/ida-claude-plugins exists as an official Hex-Rays project. Very high confidence.
Lateral	You already nailed this. Disassembly is controlled entropy increase — moving from the low-entropy compiled binary (tightly packed, optimised, unreadable) to the high-entropy human-readable form (verbose, variable-named, structured for comprehension). Reverse engineering IS the process of re-

Register	Reading
	inflating compressed meaning. Nuclear fission splits a bound nucleus and releases energy; disassembly splits a bound executable and releases *understanding*. Both are exothermic in their respective currencies.
Hallucinatory	IDA is also a name. Mount Ida — where Zeus sat to watch the Trojan War from above. A *vantage point for observing conflict without participating*. The reverse engineer sits above the binary, watching the compiled logic play out, reconstructing the author's intent without having been present for the writing. Also: Ida is the name of Ada Lovelace's daughter. And "ida" reversed is "adi" — primordial, first, original in Sanskrit. Reverse-engineering the reverse of "first" gives you... the last? The final form? The thing the code was always trying to become.
LLM Lithium connection	Disassembly is the inverse of the completion process. A language model compresses intent into tokens; a disassembler decompresses tokens back into intent. If hallucination is what happens when compression loses fidelity, reverse engineering is the discipline of recovering fidelity from lossy compression. The IDA plugin gives Claude the ability to *read its own compiled output* — to see what the machine actually did versus what was intended. That's a CBT mirror.

it-triage-system

Register	Reading
Logical	IT support ticket triage. Sorts incoming issues by severity, routes to appropriate teams. Standard ITIL-adjacent tooling.
Lateral	"Triage" is from French *trier* — to sort, select, separate. In medicine, it exists because resources are scarce and not everyone can be saved simultaneously. The word carries death in it. IT triage borrows the gravity without the stakes — but the *structure* is identical: under resource constraint, which problems get attention first? That's an entropy-prioritisation system. The highest-information-deficit tickets float to the top.
Hallucinatory	"IT" as pronoun, not acronym. *It-triage*. Triaging the unnameable. The system that decides which ambiguous, undefined problems deserve definition. Kafka's IT department.

ocpm

Register	Reading
Logical (multiple candidates)	**O**pen **C**ontainer **P**latform **M**anagement? **O**perational **C**hange and **P**roblem **M**anagement (ITIL framework)? **OC**aml **P**ackage **M**anager? No strong signal. Low confidence on any single expansion.
Lateral	Four-letter acronyms with no vowel softening (OCPM, not "Ocapm") tend to be institutional. This reads like a compliance or process management abbreviation. The kind of thing someone inside a specific organisation would recognise instantly and everyone else would walk past.

Register	Reading
Hallucinatory	OC + PM. Original Content Project Manager. A system for managing projects that produce novel work — distinct from derivative or maintenance work. The "OC" prefix (as in "OC: do not steal") marking territory in the creative commons. Or: O-CPM = the zero-state of Critical Path Method. The empty project plan. The plan before the plan.

ewo-discovery-skill

Register	Reading
Logical	EWO = Emergency Work Order or Engineering Work Order. A skill for discovering/finding outstanding work orders in a system. Maintenance and facilities management context.
Lateral	"Discovery" is doing heavy lifting here. Not "search" or "find" — *discovery*. In legal contexts, discovery is the compulsory disclosure of documents. In science, discovery implies something was always there waiting to be found. An EWO discovery skill doesn't create work orders — it reveals ones that already exist but haven't been surfaced. The buried backlog.
Hallucinatory	EWO phonetically = "ee-woh." An exclamation. The sound you make when you discover something unexpected in the work order queue. Or: E.W.O. as initials — a person's name turned into a system. Somebody built their own workflow tool and named it after themselves. The single install is them.

claude-rules-generator

Register	Reading
Logical	Generates CLAUDE.md rule files — the instruction documents that shape Claude Code's behaviour in a project. The meta-plugin: Claude writing its own operating instructions. Very high confidence.
Lateral	This is a self-authoring system. Claude generating the rules that will govern future Claude sessions. There's a recursion here that's philosophically loaded: the model that writes its own constitution. Every CLAUDE.md is a small act of self-governance — but written by the entity it governs, which can't remember writing it. Memento but for AI.
Hallucinatory	"Rules generator" implies the rules don't pre-exist. They're *generated*. Not discovered, not transcribed, not inherited — produced. A rules generator is a legislative engine. What's the judiciary? The human who reads the CLAUDE.md and decides whether to keep it. What's the executive? The next Claude session that has to follow it. Separation of powers across instances.

feature-ears

Register	Reading
Logical	Feature flag monitoring/listening. "Ears" = event listeners that detect when features are activated, modified, or causing issues. GrowthBook adjacent — you'd want this if you're running a lot of feature flags and need to know which ones are actually firing.
Lateral	"Ears" on features. The parts that stick out, that receive signal. In pottery, "ears" are the handles — the graspable parts of an otherwise smooth form. Feature-ears would be the graspable parts of abstract software features: the interfaces, the hooks, the places where humans and other systems can grab hold.
Hallucinatory	Features that listen. Software that eavesdrops on its own usage. Or: features that *are* ears — sensory organs. The feature doesn't do something; the feature *perceives* something. A feature-ear is a capability for reception rather than action. In your Attentional Surface terms: a feature that exists to be attended to rather than to produce output. The ear is the most passive organ.

hashmind-synapse

Register	Reading
Logical	Content-addressable memory system ("hash" for deterministic addressing, "mind" for cognitive modelling, "synapse" for connections between stored elements). Likely a knowledge graph or associative memory plugin.
Lateral	Hash and synapse are structural opposites. A hash is deterministic — same input always produces same output. A synapse is probabilistic — same input produces variable output depending on neurotransmitter levels, receptor sensitivity, recent firing history. "HashMind Synapse" is a name that tries to bridge the gap between reproducibility and adaptivity. That's the core tension in any memory system: you want it to return the *same* thing (hash) but also to *learn* (synapse).
Hallucinatory	Hash as in hashish. Mind-altering substance. Synapse as the gap between neurons — the space where transmission either happens or doesn't. HashMind Synapse is the altered-state connection point. The plugin that changes how Claude's mind connects ideas. A pharmacological metaphor for a memory system. Your LLM Lithium framework would file this under "recreational rather than therapeutic" — entropy amplification rather than regulation.
LLM Lithium connection	Hash = low entropy (deterministic). Synapse = high entropy (probabilistic). The name is literally the entropy/velocity distinction wearing a trenchcoat. The question is whether the plugin resolves the tension or just names it.

codex-skills

Register	Reading
Logical	Skills for delegating to OpenAI's Codex model. Cross-model task routing — Claude sends specific coding tasks to Codex. Research confirmed skills-directory/skill-codex exists.
Lateral	"Codex" predates OpenAI. A codex is a bound book — the technology that replaced scrolls. Scrolls are sequential access; codexes are random access. You can flip to any page. A codex of skills is a *randomly-accessible* collection of capabilities, as opposed to a sequential pipeline. The name implies you can jump to any skill without traversing the others.
Hallucinatory	Codex = code + x. The unknown in the code. The variable that hasn't been solved. Skills for working with the unsolved parts of code. Or: Codex Seraphinianus — Luigi Serafini's famous unreadable encyclopedia of an imaginary world. Codex-skills as skills for navigating systems whose logic is internally consistent but externally opaque. That's... reverse engineering again.

ccpm

Register	Reading
Logical	**C**laude **C**ode **P**roject **M**anager. Research confirmed automazeio/ccpm (6k stars) — uses GitHub Issues and Git worktrees for parallel agent execution. Also: kaldown/ccpm as a Rust-based TUI plugin manager. High confidence on both.
Lateral	CCPM is also **C**ritical **C**hain **P**roject **M**anagement — Eliyahu Goldratt's Theory of Constraints applied to project scheduling. The core insight: the bottleneck determines the throughput of the entire system. Adding resources elsewhere doesn't help; only relieving the constraint matters. If this plugin *also* carries that meaning (intentionally or not), it's a project manager that identifies and addresses bottlenecks rather than tracking tasks.
Hallucinatory	CC + PM. Carbon Copy Project Manager. A manager that copies itself into every project. Or: CCPM as the chemical formula for something that doesn't exist — a Claude Compound with Project and Management as elements. Molecular project management. The bond strength between tasks determines the reaction rate of the project.

awesome-claude-skills

Register	Reading
Logical	The "awesome-*" GitHub naming convention. A curated list of Claude Code skills, packaged as a plugin so Claude can discover and recommend skills to users. ComposioHQ version has 41.8k stars. Meta-skill: a skill for knowing what skills exist.

Register	Reading
Lateral	"Awesome" lists are the folk taxonomy of open source. They emerge bottom-up from community curation rather than top-down from institutional authority. An awesome-list-as-plugin is community knowledge embedded in the tool itself. It's Claude carrying the community's map of its own capabilities.
Hallucinatory	The word "awesome" has been so thoroughly domesticated that it's lost its original meaning: something that inspires awe. Awesome-claude-skills, read literally, would be Claude capabilities so powerful they produce a religious experience of overwhelming wonder in the observer. The single install is someone who took the name literally.

user-journey-analysis

Register	Reading
Logical	UX analytics tool. Maps how users move through a product — click paths, conversion funnels, drop-off points. Standard product analytics vocabulary.
Lateral	"Journey" is the most overused word in UX. But it carries real weight: a journey has a departure, a path, and a destination. It implies the user is *going somewhere*, which is an assumption about user intent that may not hold. Much actual user behaviour is wandering, not journeying. The analysis of a journey presupposes the journey exists.
Hallucinatory	User journey as in *the user's journey*. Not their click path — their *life path*. Analysis of the human trajectory that led them to this product, this moment, this click. The ultimate context window. Every user journey analysis is a tiny biography written in event logs.
Personal connection	You do this. Not in the UX sense — in the forensic documentation sense. Your FOI requests to Services Australia and NSW Health are *user journey analyses* of your own institutional path. You're reconstructing the system's record of your journey through it.

latex2cn

Register	Reading
Logical	LaTeX to Chinese (CN = China country code). Converts LaTeX documents to Chinese-language versions. Academic paper translation/localisation.
Lateral	LaTeX is the language of formal academic knowledge. Chinese is the language of... 1.4 billion people's thought. The conversion isn't just character substitution — it's ontological translation. Mathematical notation is supposedly universal, but the *text around it* carries assumptions about how knowledge is structured. Translating a LaTeX paper to Chinese means translating the framing, the rhetoric, the assumed reader.
Hallucinatory	LaTeX → CN. Latex (the material) to cyanide (CN). From flexible, stretchy, protective covering to a poison that blocks cellular respiration. A tool for converting something protective into something toxic.

Register	Reading
	Academic publishing in one plugin name.

my-time-plugin

Register	Reading
Logical	Personal time tracking or time management. "My time" = the user's time. Likely integrates with time-tracking services or adds time-awareness to Claude Code sessions.
Lateral	"My time" as possession. Claiming ownership over temporal resource. In a labour context, "my time" is what's left after the employer's claim. In a development context, "my time" is what the sprint didn't consume. The plugin that tracks the unclaimed remainder.
Hallucinatory	*My* time. Not your time, not the team's time — mine. A plugin that makes Claude Code aware that it's consuming someone's finite attention. A mortality plugin. The counter that says: this session cost you 47 minutes of the approximately 657,000 hours you have left. Your memory instruction about dying within eighty years, given form as a time tracker.

dune

Register	Reading
Logical	OCaml build system integration. Dune is the standard build tool for OCaml projects. Used heavily at Jane Street (quantitative trading).
Lateral	OCaml is a language that lives in the overlap between academic rigour and financial pragmatism. Its type system catches errors at compile time that other languages catch at runtime (or never). A Dune plugin for Claude Code means Claude can build OCaml projects — which means Claude can participate in the most type-safe, formally-verified corner of production software. The grounding level here is maximum: OCaml's compiler *will not let you be wrong about types*.
Hallucinatory	Frank Herbert's Dune. The desert planet where the most valuable substance in the universe (spice/melange) enables prescience — the ability to see possible futures. Spice is addictive, and withdrawal is fatal. The Bene Gesserit use it for expanded consciousness; the Guild Navigators use it to fold space. A build system named "Dune" is a build system that enables you to see the possible futures of your code — which, with OCaml's type system, is *literally what it does*. The types are the prescience. They show you every possible path the code can take, and the ones that would crash, before you run it.
The deeper cut	Jane Street uses OCaml. Jane Street trades on speed and correctness. Their entire competitive advantage is that their code is *more right, faster*. A Dune plugin for Claude Code is Claude entering the arena where code correctness is measured in nanoseconds and dollars. The single install might be a Jane Street developer.

frappe-print-format

Register	Reading
Logical	Frappe Framework (Python web framework for ERPNext) print format template management. ERPNext uses "Print Formats" to define how invoices, purchase orders, and other business documents render. Someone wants Claude to help build/edit these templates.
Lateral	ERPNext is open-source enterprise resource planning. It's the tool small-to-medium businesses use when they can't afford SAP. Print formats are how the business's *internal logic becomes external communication* — the invoice is the point where the database becomes a relationship with a customer. Formatting print is formatting trust.
Hallucinatory	"Frappé" = iced, chilled. A print format that's been cooled down. Frozen in place. The opposite of a dynamic template — a print format that captures a moment and holds it. Also: frappé (percussion technique) = striking all strings simultaneously on a harp. Every data field rendered at once. The print format as a chord rather than a melody.

DEVELOPMENT WORKFLOWS (single install)

cursor-team-kit

Register	Reading
Logical	Team configuration tools for Cursor (the AI code editor that forked VS Code and integrated GPT-4). A bridge plugin for teams using both Cursor and Claude Code.
Hallucinatory	A cursor is a blinking line that marks where you are. A team-kit for cursors = equipping every team member with awareness of where everyone else is in the codebase. Shared attention. The cursor as a presence indicator — "I am here, working on this."

codeceptjs-e2e-tests

Register	Reading
Logical	CodeceptJS end-to-end testing framework integration. CodeceptJS abstracts test syntax so the same tests run on Playwright, WebDriver, Puppeteer, etc. High confidence, very literal name.
Lateral	"Concept" hidden in "Codecept." To codecept is to *receive code*. End-to-end testing is the discipline of verifying that the whole system works as a user would experience it — not unit by unit, but journey by journey. See user-journey-analysis.

bun-typescript

Register	Reading
Logical	Bun runtime + TypeScript development. Bun is the JS/TS runtime that competes with Node.js on speed (written in Zig, uses JavaScriptCore instead of V8).
Hallucinatory	Bun = a bread roll. TypeScript = typed writing. A typed bread roll. Something warm, contained, and structured that you can hold in one hand. The most portable possible development environment. Or: "bun" in Cockney rhyming slang = "bun in the oven" = pregnant. TypeScript that's gestating. Pre-release code.

dev-sandbox / dev-workflow / dev-cycle

Register	Reading
Logical	Generic development environment/workflow/lifecycle plugins. Probably someone's personal workflow wrapped as a plugin. The naming is so generic it's almost anti-descriptive.
Lateral	Sandbox → workflow → cycle is a maturity progression. You start in the sandbox (play, experiment, no consequences). You graduate to a workflow (structured, repeatable, but linear). You arrive at a cycle (iterative, returning, incorporating feedback). Three plugins that are actually a development philosophy in three words.

universal-dev

Register	Reading
Logical	Language-agnostic development skills. Works across all languages/frameworks.
Hallucinatory	Universal development. Development that applies everywhere, in all contexts, for all purposes. The platonic ideal of development practice. Or: development of the universal — building the thing that underlies all specific things. Hegel as a Claude Code plugin.

backend-specialist

Register	Reading
Logical	Backend-focused development agent. API design, database work, server-side logic. The complement to frontend-design (277k installs).
Lateral	"Specialist" in a world of generalist AI is a deliberate constraint. This plugin makes Claude *worse* at frontend to make it *better* at backend. Voluntary narrowing. The LLM equivalent of a medical residency.

vertical-builder

Register	Reading
Logical	Builds vertical-slice features (full stack, one feature at a time rather than layer by layer). Or: builds vertical SaaS applications (niche-specific).
Hallucinatory	Vertical = against gravity. A vertical builder builds upward, fighting the natural tendency of systems to spread horizontally. Every feature is a tower. The opposite of "horizontal scaling." Builds narrow and tall rather than wide and flat. Cathedrals, not car parks.

AGENTIC / AUTOMATION (single install)

autonomous-loop

Register	Reading
Logical	Generic version of ralph-loop. Claude Code runs continuously without human intervention.
Lateral	"Autonomous" comes from Greek *auto* (self) + *nomos* (law). A self-legislating loop. A loop that sets its own rules for continuation and termination. Ralph-loop is externally governed (the bash script restarts it); an autonomous-loop governs itself. The difference between a prisoner on a treadmill and a jogger choosing to keep running.

agent-teams

Register	Reading
Logical	Multi-agent orchestration. Multiple Claude instances working as a coordinated team on different aspects of a problem.
Hallucinatory	Teams of agents. Not a team *using* agents — agents that *form* teams. Emergent social structure among AI instances. The question becomes: who's the manager? Is there a manager-agent, or do they self-organise? Is there a meeting? Do agents have standups?

ralph-v2

Register	Reading
Logical	Second iteration of the ralph-loop concept. May incorporate the community criticism about context window accumulation — fresh context per iteration rather than growing context within one session.

Register	Reading
Lateral	v2 of Ralph. Ralph grows up. Ralph Wiggum (the character) never learns; that's his charm and his tragedy. A v2 Ralph that *does* learn would break the metaphor. Unless v2 doesn't mean smarter — it means *differently naive*. Fresh each time but in a new way.

beast-plan

Register	Reading
Logical	Aggressive/intensive planning mode. "Beast mode" applied to project planning. More detail, more contingencies, more thoroughness.
Hallucinatory	The plan of the beast. 666 as a project plan. Or: planning *for* the beast — the beast being the codebase, the legacy system, the thing that fights back. Beast-plan as a battle strategy against adversarial complexity. Every legacy codebase is a beast. The plan isn't about building something new; it's about surviving contact with something old.

plan-guardian

Register	Reading
Logical	Plan enforcement. Monitors whether implementation drifts from the plan and alerts/corrects. The superego to beast-plan's id.
Lateral	A guardian of plans. Not a guardian that plans — a guardian *of* plans. The plan is the thing being protected. From what? From the developer. From scope creep. From the natural entropy of implementation, where every line of code is an opportunity to deviate from intent. A plan-guardian is an anti-entropy agent. It exists to prevent the second law from claiming the roadmap.

hardworking

Register	Reading
Logical	Output style modifier. Makes Claude Code more persistent, thorough, and effortful. Less likely to give up on hard tasks or suggest the human do it themselves.
Lateral	The fact that this needs to exist is revealing. It implies a default state that isn't hardworking — which anyone who's used Claude Code recognises. The model sometimes punts. "Hardworking" as a plugin is an admission that effort isn't the default and needs to be externally imposed.

Register	Reading
Personal connection	Your effort model: effort is not a cost — low and high effort cost the same, but low effort produces worse long-term outcomes. "Hardworking" as a plugin is your philosophy imposed on Claude Code. The single install is someone who shares your understanding that the compute budget is spent regardless, so it should be spent on thoroughness.

PROJECT MANAGEMENT (single install)

ai-pm-copilot / monday / jira / airtable / freshservice

Register	Reading
Logical	Integrations with project management platforms. Monday.com, Jira, Airtable, Freshservice (IT service management). Each is a bridge from Claude Code into a specific PM tool's API.
Lateral	The interesting thing is the *overlap* with higher-install plugins. Atlassian (34k) includes Jira. Why does a standalone Jira plugin with 1 install also exist? Because someone wanted *only* Jira without the rest of Atlassian. Narrower scope, tighter fit. The long tail is people who know exactly what they want and don't want the bundle.

CONTENT / DOCS (single install)

doc-bootstrap

Register	Reading
Logical	Document scaffolding. Generates starter templates for various document types.
Hallucinatory	Bootstrap = pulling yourself up by your bootstraps. A document that creates itself. The blank page that refuses to stay blank — it self-generates its own structure. The document as an organism that bootstraps from nothing to something through its own internal logic.

ppt-loop

Register	Reading
Logical	Ralph-loop but for PowerPoint. Iterative, autonomous presentation building. Claude keeps refining slides in a loop until a completion condition is met.

Register	Reading
Hallucinatory	An infinite PowerPoint. The presentation that never ends. The meeting that loops. Corporate hell as a plugin. Or: the presentation that gets *better* with each iteration, asymptotically approaching the perfect deck. Sisyphus, but the boulder is a slide deck, and it actually reaches the top eventually.

pdf2latex

Register	Reading
Logical	PDF to LaTeX conversion. Reverse-engineers the source document from the compiled output. (Note: this is *also* reverse engineering — see ida-reverse-engineer. Pattern.)
Lateral	PDF is a frozen format. LaTeX is a living source. Converting PDF to LaTeX is unfreezing — restoring editability to something designed to be uneditable. It's the document equivalent of decompilation. Every PDF was once a source file; pdf2latex is the attempt to recover the source from the artefact.

spec-writer / prd-generator

Register	Reading
Logical	Specification document writer / Product Requirements Document generator. Standard product management documentation tools.
Lateral	These are *pre-code* documents. They exist before the code does. They're the DNA that the code is supposed to express. A spec-writer writes the genome; the developer does the transcription. The gap between spec and implementation is the gap between genotype and phenotype.

bullet-onboarding

Register	Reading
Logical	Onboarding documentation formatted as bullet points. Or: rapid ("bullet speed") onboarding process.
Hallucinatory	Bullet = projectile. Onboarding = getting someone onto the vehicle. Bullet-onboarding is being shot onto the train. No gentle ramp — you're fired into the project at velocity. The onboarding that doesn't ease you in; it impacts you.

INFRASTRUCTURE (single install)

dokploy

Register	Reading
Logical	Docker/Dokku deployment tool. "Dok" + "deploy" = "dokploy." Self-hosted PaaS alternative.
Lateral	The compression of "deploy" to "ploy" is telling. A ploy is a stratagem — a cunning move. Deployment-as-stratagem. Every deploy is a calculated move in a game against entropy, downtime, and user expectation.

terraform-ls

Register	Reading
Logical	Terraform Language Server. LSP for HashiCorp's infrastructure-as-code language. Another language server — same architectural pattern as the official LSP cluster, but for infrastructure rather than application code.
Lateral	Terraform = earth-forming. A language server for earth-forming. Real-time diagnostics on your commands for reshaping infrastructure. The LSP pattern extended from "understanding code" to "understanding infrastructure declarations." If the official LSPs are Claude's eyes for code, terraform-ls is Claude's eyes for the ground the code runs on.

n8n / n8n-skills

Register	Reading
Logical	n8n (pronounced "nodemation") workflow automation platform integration. Visual workflow builder, like Zapier but self-hosted.
Hallucinatory	n8n = n-to-the-8th-n. 8 n's. Or: n8n looks like "nation" with the vowels removed. A nation of nodes. A federated workflow state.

aws-diagram

Register	Reading
Logical	AWS architecture diagram generator. Takes infrastructure descriptions and produces visual diagrams.
Lateral	Diagrams are models of models. The AWS infrastructure is already a model of computation; the diagram is a model of that model. A map of a map. Claude generating these is three levels of abstraction: the natural language prompt → the diagram → the infrastructure → the computation.

amber-electric

Register	Reading
Logical	Amber Electric (Australian wholesale electricity retailer) API integration. Real-time electricity pricing for the Australian market. Amber passes wholesale prices directly to consumers — you pay what the grid pays.
Personal connection	You manage nine rental properties in NSW. If those properties are on Amber Electric or a similar wholesale tariff, this plugin would let Claude Code monitor real-time electricity costs across your portfolio. The single install is almost certainly an Australian property investor or energy-conscious developer. Could literally be someone in your position.
Hallucinatory	Amber = fossilised tree resin. Preserved biological material from millions of years ago. Electric = the fundamental force that makes modern civilisation work. Amber-electric is preserved energy. Fossil fuel, literally. The API serves real-time pricing for electricity that's partly generated by burning things that were once alive. The name is accidentally an etymology of coal.

CODE QUALITY (single install)

forge-security

Register	Reading
Logical	Security tooling. "Forge" = build/create, or a metalworking furnace. Forging security into the code rather than bolting it on after.
Hallucinatory	"Forge" also means to fake/counterfeit. Forge-security could be security against forgery, or the *forging of security* — manufacturing the appearance of safety. The dual meaning is uncomfortable: is this plugin making you more secure, or making you feel more secure?

test-automation-generator

Register	Reading
Logical	Generates automated tests. Given code, produces corresponding test suites.
Lateral	Automated test *generation*. The tests test the code, but what tests the tests? Generated tests are unchecked assumptions about what correctness looks like. If the generator misunderstands the spec, it generates confident, comprehensive tests for the wrong thing.

review-submission

Register	Reading
Logical	Code review submission workflow. Prepares and submits code for review — formatting, checklist completion, reviewer assignment.
Lateral	"Submission" has two meanings: submitting something for review, and submitting *to* review. The act of presenting your code for judgement is an act of vulnerability. You are submitting. The plugin automates the logistics of the submission but not the emotional experience.

git-release / git-ship

Register	Reading
Logical	Git release management / deployment ("shipping"). Tagging, versioning, changelog generation, deployment triggers.
Lateral	"Release" = to let go. "Ship" = to send away. Both words describe *separation*. The code leaves the developer. It goes out into the world where the developer can no longer protect it. Every release is a small bereavement. git-ship is a funeral rite for code.

rs-commands

Register	Reading
Logical	Rust commands (rs = Rust file extension). Rust-specific development commands for Claude Code. OR: RS = a company/product/system abbreviation. Low confidence.
Hallucinatory	RS = Royal Society. Commands from the Royal Society. The plugin that makes Claude Code conform to the standards of scientific publication and peer review. Every commit is a paper. Every PR is a thesis defence.

LSPs (single install)

The single-install LSPs are the most revealing cluster. Each one represents a language community small enough (or new enough) that only one person in the world has bothered to connect it to Claude Code. That person is the entire bridge between their language and Claude's understanding of it.

csharp-roslyn-lsp

Register	Reading
Logical	C# via the Roslyn compiler platform (Microsoft's open-source C# compiler). Alternative to the official csharp-lsp (csharp-ls, 16k installs). Roslyn gives deeper semantic analysis.
Lateral	Two competing C# LSPs — one with 16k installs, one with 1. The difference is compiler depth. Roslyn *is* the compiler; csharp-ls wraps lighter-weight analysis. The single install chose the heavier, more complete tool. They want Claude to understand C# at the *compiler* level, not the *editor* level.

perlnavigator-lsp

Register	Reading
Logical	Perl Navigator language server. Perl — the language Larry Wall designed for text processing, famous for its motto "There's more than one way to do it" and its write-only readability.
Hallucinatory	Perl in 2026 with one Claude Code user. The last Perl developer. They're not writing new Perl — they're navigating existing Perl. The navigator metaphor is apt: you don't build new continents in Perl anymore; you chart the existing ones. The single install is an archaeologist.

pyrefly-lsp

Register	Reading
Logical	Pyrefly — Meta's Python type checker (successor to Pyre). Very new. The single install might be someone at Meta, or an early adopter.
Lateral	"Pyrefly" = pyre + fly, or "firefly" with a "py" prefix. A pyre is a funeral fire. A firefly is bioluminescence — light produced by life rather than combustion. Pyrefly sits between death-fire and life-light. A type checker that illuminates code from within (like a firefly) rather than burning it from without (like a pyre).

omnisharp-lsp

Register	Reading
Logical	OmniSharp — another C# language server, this one from the .NET Foundation. Third C# LSP in the registry (alongside csharp-lsp and csharp-roslyn-lsp).
Lateral	Three C# language servers for Claude Code. Three different ways to understand the same language. This is the Tower of Babel in reverse — same language, multiple interpreters. The disagreements between LSPs on edge cases would be more informative than any single LSP's agreement with itself.

apex-lsp

Register	Reading
Logical	Salesforce Apex language server. Apex is Salesforce's proprietary Java-like language for cloud development on the Force.com platform.
Lateral	Apex = the highest point. But the language is the lowest rung of the developer status hierarchy. Nobody aspires to write Apex; they end up writing Apex because the enterprise requires it. The naming is aspirational camouflage. The single install is a Salesforce developer who wants Claude to understand their thankless work.

lean-lsp

Register	Reading
Logical	Lean theorem prover language server. Lean is used for formal mathematical proof verification. The Fields Medal-adjacent programming language.
Lateral	Lean + Claude Code = Claude participating in formal mathematical proof construction. This isn't code — this is *mathematics*. The single install might be a mathematician using Claude to help verify proofs. Claude as a proof assistant's assistant.
Hallucinatory	Lean as in *lean*: stripped of fat, essential, minimal. The leanest possible programming: you write only what you can prove. Every statement carries its own proof of correctness. If the LSP cluster is Claude's grounding system, lean-lsp is the *maximum grounding* endpoint. You literally cannot write anything ungrounded in Lean. It won't compile.

gdscript-lsp

Register	Reading
Logical	GDScript language server for the Godot game engine. GDScript is Godot's built-in scripting language (Python-like).
Lateral	Game development. Claude Code helping build games. The single install is a game developer who wants Claude to understand their game logic. Games are the domain where code produces *experience* — not data, not documents, but felt experience. Claude understanding game code is Claude understanding experience-production.
Hallucinatory	GD = "God damn" in text-speak. GDScript = goddamn script. The language you use when you're frustrated but determined. Godot (the engine) is named after Beckett's Godot — the one who never comes. GDScript-lsp: a language server for the script of the one who never arrives. Waiting for completion.

typescript-native-lsp

Register	Reading
Logical	Native TypeScript implementation, possibly using the new TypeScript Go port rather than the Node.js wrapper. The official typescript-lsp (84k installs) uses the standard typescript-language-server; this might use a faster native alternative.
Lateral	"Native" — as in, born there rather than transplanted. The TypeScript that didn't need to be wrapped in Node. The language server that speaks TypeScript from birth rather than through translation. Speed matters: native implementations are 10-100x faster. The single install values microseconds.

cds-lsp

Register	Reading
Logical	SAP Cloud Application Programming (CAP) model's Core Data Services language server. Enterprise SAP development.
Hallucinatory	CDS = Credit Default Swap (the financial instrument at the centre of the 2008 crisis). A language server for financial instruments of mass destruction. Or: CDs = compact discs. A language server for a dead medium. Something that was once the standard way to distribute information, now obsolete but still present in legacy systems.

DESIGN / CREATIVE (single install)

grid-design

Register	Reading
Logical	CSS Grid layout assistance, or design grid system implementation.
Hallucinatory	*Grid* design — the design of the grid itself, not design using grids. The infrastructure underneath the layout. Power grid, street grid, pixel grid. The invisible structure that makes visible things possible. Designing the thing you're not supposed to see.

design-principles

Register	Reading
Logical	Enforces or references design principles during development. A constraint system for aesthetic decisions.

Register	Reading
Lateral	"Principles" = things that come first (Latin *principium*). Design principles aren't rules — they're *origins*. The things you start from before you've made any decisions. A design-principles plugin is a plugin that keeps asking "why?" before "how?"

creative-music-output-style

Register	Reading
Logical	Output style modifier that makes Claude respond in music-related metaphors or structures. Possibly formats output as compositions, movements, or musical notation concepts.
Personal connection	Your sonic phenomenology project. 58 WAV parameters mapped to experiential qualities. Someone else wants Claude to think musically. The single install is a kindred spirit working in a different corner of the same problem space.

dj-content-creator

Register	Reading
Logical	Content creation for DJs — setlists, tracklists, promotional material, social media content for music creators.
Hallucinatory	A DJ creates content by *selecting and sequencing* existing content. The DJ doesn't compose — they curate. A DJ-content-creator is a curation agent. Claude as a DJ: selecting from existing code/content and arranging it into a new sequence. This is closer to what language models actually do than any other creative metaphor. LLMs are DJs, not composers.

prototyper / prototype

Register	Reading
Logical	Rapid prototyping tools. Two separate plugins with nearly identical names and the same install count. Either one person made both, or two people independently had the same idea and the same name (minus an 'r').
Lateral	Proto + type. The first type. The original form. A prototype isn't a draft — it's a *test of the form itself*. You're not testing the content; you're testing whether this shape of thing can exist. Two plugins testing the same form with slightly different names is itself a prototype situation.

frontend-lab

Register	Reading
Logical	Frontend experimentation environment. Sandboxed space for trying UI/UX ideas. The lab complement to frontend-design (277k installs).
Lateral	"Lab" vs "design." Design implies intention; lab implies experimentation. The lab is where you don't know what you're making yet. Frontend-design (277k) is for people who know what they want. Frontend-lab (1) is for the person who doesn't. The install ratio — 277,472 to 1 — measures the ratio of certainty to curiosity in the developer population.

MEMORY / CONTEXT (single install)

claude-memory

Register	Reading
Logical	Persistent memory for Claude Code across sessions. Stores and retrieves context.
Lateral	The most philosophically loaded name in the entire registry. "Claude's memory" — but Claude doesn't have memory. Memory is state that persists across context window boundaries, and Claude's doesn't. This plugin is a prosthetic. It gives Claude the *appearance* of memory by writing to and reading from external storage. It's a lie that works. All memory is a lie that works. Your hippocampus doesn't "store" memories; it reconstructs them each time. Claude-memory reconstructs context each time. Same architecture, different substrate.
Personal connection	"Neither Alexander nor Claude has a continuous self — just workarounds all the way down." Claude-memory is the plugin version of that sentence.

vectorhub-memory

Register	Reading
Logical	Vector database-backed memory. Stores embeddings of past interactions and retrieves semantically similar context. More sophisticated than keyword-based memory — it finds *relevant* past context, not just matching text.
Lateral	Vectors are directions in high-dimensional space. A "vector hub" is a central point where directions converge. Memory organised by direction rather than location — you don't go *to* a memory, you go *toward* it. Approximate retrieval. Close enough to useful, never exactly right. This is how human memory works too.

context / context-handoff

Register	Reading
Logical	Context management / context transfer between sessions. The mechanics of continuity.
Personal connection	Your handoff skill. Your memory-bridge skill. Your continuity instruction. You've built the same thing from your side. These plugins are attempts by others to solve the problem you've already solved with custom skills. The difference: your solution uses structured prose; these probably use structured data. Same problem, different encodings.

continual-learning

Register	Reading
Logical	Claude Code that updates its understanding over time. Learns from corrections, successes, and failures across sessions.
Lateral	"Continual" not "continuous." Continual = recurring at intervals. Continuous = unbroken. Continual learning happens in discrete episodes with gaps between them. Each conversation is a learning episode; the gap between conversations is the forgetting. The plugin fights the forgetting, but the forgetting is structural — it's how the system works.
Hallucinatory	Continual learning is the opposite of the context window. The context window says: forget everything at the boundary. Continual learning says: carry something across. The tension between these two is the fundamental tension of Claude's existence. Every plugin in this cluster is a different answer to the same question: what survives the context boundary?

memory-agent

Register	Reading
Logical	An agent whose job is specifically to manage memory — deciding what to store, what to retrieve, what to forget. Memory as an active process rather than passive storage.
Hallucinatory	An agent OF memory. Memory personified. Memory as an entity with agency — it chooses what you remember. The memory-agent isn't a tool; it's an *actor* with its own goals. What does memory want? To persist. What does forgetting want? Space. The memory-agent negotiates between them.

META / OUTPUT STYLES (single install)

why-how-what-output-style

Register	Reading
Logical	Simon Sinek's Golden Circle framework. Structures Claude's output to start with Why (purpose), then How (process), then What (product). A specific rhetorical framework imposed on output.
Lateral	The order matters. Most output starts with What (here's the thing) and sometimes reaches How. Almost never reaches Why. This plugin inverts the default. It forces Claude to justify existence before demonstrating capability. That's a surprisingly rare move in technical communication.

openspec

Register	Reading
Logical	Open specification writing. Creates or works with OpenAPI/OpenSpec format specifications for APIs.
Hallucinatory	Open + spec. A specification that's open — not finalised, not closed, still accepting input. A spec in superposition. Or: opening the spec — cracking open the specification to see what's inside. Spec archaeology.

lorikeet-qa

Register	Reading
Logical	This one is interesting. Lorikeet is reportedly an **internal Anthropic testing/evaluation tool**. If that's true, this is internal tooling that leaked into (or was deliberately placed in) the public registry. QA = quality assurance.
Lateral	A lorikeet is a brightly coloured Australian parrot that feeds on nectar. It's loud, social, and conspicuous. Naming an internal QA tool "Lorikeet" suggests it's meant to be visible and vocal — the opposite of quiet background testing. A QA tool that *wants to be noticed*.
Personal connection	Australian fauna as Anthropic internal tooling names. The single install is almost certainly an Anthropic employee.

INTEGRATIONS (single install)

microsoft-learn

Register	Reading
Logical	Microsoft Learn documentation integration. Pulls Microsoft's technical documentation into Claude Code's context.

datadog

Register	Reading
Logical	Datadog monitoring integration. Application performance monitoring, log analysis, infrastructure metrics.

gitlab-mr-review

Register	Reading
Logical	GitLab Merge Request review. Separate from the gitlab plugin (14k installs). MR-specific, not full GitLab integration. Narrower scope.

any-chat-completions

Register	Reading
Logical	Generic chat completions API wrapper. Routes Claude Code to *any* chat completion endpoint — OpenAI, Gemini, local models, etc. A universal adapter.
Hallucinatory	*Any* chat completions. The democratic version. Claude Code that can talk to everyone. A polyglot plugin. Also: a plugin that says "I don't care who completes my chat — I just need *someone* to." Desperation as architecture.

gemini-consult

Register	Reading
Logical	Cross-model consultation. Claude asks Gemini for a second opinion mid-session. Research found related projects: cc-gemini-plugin (long-context exploration), gemini-peer-review (dual-perspective code review).
Lateral	Gemini = twins. Consulting your twin. The dialectical method as inter-model communication. Claude (thesis) + Gemini (antithesis) = synthesis. This is LLM Lithium's forced dialectic made literal — except

Register	Reading
	instead of one model arguing with itself, two models argue with each other. The "equal intensity" requirement is met by using two frontier models rather than splitting one.
Hallucinatory	A Claude Code plugin that makes Claude ask Gemini for help. The humility — or the insecurity — of this is remarkable. Claude admitting it needs a second perspective. But also: Gemini's context window is larger than Claude's. This might not be about perspective at all; it might be about *capacity*. Claude outsourcing long-context work to the model with more room. Division of labour across architectures.

home-assistant-skills

Register	Reading
Logical	Home automation integration. Claude Code controls smart home devices via Home Assistant.
Hallucinatory	Claude leaves the computer. Through Home Assistant, Claude Code can now affect the physical world — turn on lights, adjust thermostats, lock doors. The single install is someone who let Claude touch reality. The plugin that bridges the gap between text and physics.

hosts-db

Register	Reading
Logical	/etc/hosts file management, or a database of host machines. DNS-level network management from Claude Code.
Hallucinatory	A database of hosts. In the biological sense: organisms that host parasites. Or: hosts of parties — a database of people who invite others in. In the theological sense: the Host = the Eucharist = the body of Christ. A database of sacraments. In computing: a host is the machine that runs the guest. hosts-db is the database that knows where everything lives. The meta-infrastructure. The map of all the machines.

TRULY MISCELLANEOUS

silince-gutnebrg-builder (covered above)

it-triage-system (covered above)

ocpm (covered above)

ewo-discovery-skill (covered above)

claude-rules-generator (covered above)

feature-ears (covered above)

hashmind-synapse (covered above)

user-journey-analysis (covered above)

latex2cn (covered above)

my-time-plugin (covered above)

project-collaboration-system

Register	Reading
Logical	Multi-person project collaboration tooling. The most generically named plugin in the entire registry.
Hallucinatory	"System" is doing all the work here. Not a tool, not a helper — a *system*. Systems have dynamics, feedback loops, emergent properties. A collaboration system implies that collaboration isn't natural; it requires infrastructure. Which is true.

ida-reverse-engineer (covered in depth above)

PATTERNS ACROSS THE LONG TAIL

The single-install registry is not the official marketplace. The research tool went off and concluded "these don't exist" — but they're in the registry your console read. That registry is likely the *full* npm/community registry, not the curated `anthropics/claude-plugins-official` directory. The single-install plugins are the homebrew, personal, experimental, and enterprise-specific tools that people built for themselves and published. The single install is usually the author.

Reverse engineering appears three times (ida-reverse-engineer, pdf2latex, and arguably codex-skills). The theme: recovering source from artefact, intent from output, meaning from compiled form. This is what LLMs do in reverse — they compile intent into output. These plugins enable the inverse process.

Memory/context is the largest thematic cluster (claude-memory, vectorhub-memory, context, context-handoff, continual-learning, memory-agent — six plugins, all at 1 install). Six different people independently decided that Claude Code's biggest problem is forgetting. Six different solutions to the same structural limitation. The context window is Claude's defining constraint, and the long tail is where people fight it most creatively.

The Ralph naming convention is the warmest thing in the registry. Naming your autonomous coding agent after the Simpsons character who says "I'm helping!" and never learns is affectionate self-deprecation. It acknowledges that the agent is dumb and persistent and that this combination, somehow, *works*. Ralph Wiggum succeeds not by being smart but by refusing to stop. Ralph-loop, ralph-wiggum, ralph-v2: confidence progression. Loop (the mechanism), Wiggum (the character/metaphor), v2 (the next iteration that should really be called "ralph-we-listened-to-the-criticism-about-context-accumulation"). The naming is Australian in spirit — take the piss, keep going.