Episode Book

EPISODE #6629

How Canopy Searches AI Skills and Tools on Demand

PUBLISHED	16 November 2026 at 14:43 UTC
DURATION	36:36
LISTEN	myweirdprompts.com/episode/canopy-skill-tool-search-84956/
HOSTS	Corn, Herman, Hilbert
SCRIPT MODEL	anthropic/claude-opus-5.5
VOICES	omnivoice
PIPELINE	V5.33
BOOK GENERATED	16 November 2026 at 22:22 UTC

SYNOPSIS

Nine hundred and twenty-seven tool names cost 20,010 tokens at the start of every session, even with schemas deferred. Canopy is a self-hosted AI gateway built to fix that by putting skills, plugins, tools, and standing notes behind a single searchable MCP endpoint. We look at why tool discovery exists in Anthropic's API while skill discovery stalls in Claude Code and MCP's SEP 2640, how Canopy's 14-service stack combines Tantivy BM25 and BGE-M3 vector search, and what lessons from OpenViking shaped its design.

TRANSCRIPT



Corn

Twenty thousand and ten tokens. That is what nine hundred and twenty-seven tool names were costing Daniel at the start of every session, and that was with the schemas already deferred.



Herman

Just the names. Not the parameters, not the definitions. A list of what exists, sitting in the context window before he has typed a word.



Corn

That measurement is more or less where Canopy comes from, and Canopy is what Daniel wants us to pull apart today. It's the self-hosted AI gateway he's been building for his own setup, he moved his deployment to version one point ten yesterday, and the idea is that an agent doesn't get skills, plugins, tools or his own standing notes installed. It searches for them and loads the one it needs. His argument is that tools have grown a discovery layer and skills haven't. For the clients people actually use, that holds: Claude Code still loads every skill's description into every session, and the request to give skills a search tool was closed by a bot without being built. Where it softens is the wider field, because a few projects already search skills, and Canopy's own documentation says as much. So the claim worth testing is the bundle. Skills, tools and your own notes, behind one private endpoint you can search, at a time when the clients still won't do skill discovery for you.



Herman

A far sturdier thing to defend. "Nobody has built this" dies the moment someone finds a repo. "I want these four things in one place, on my own machine, with no keys on the client" survives any search engine you throw at it.



Corn

Before the anatomy, though, the twenty thousand. If deferring hides the schemas, what's left that costs that much?



Herman

The roll call. Deferred loading means the agent sees a name and a short hint, and only pulls the full definition when it wants to call something. At forty tools that's a rounding error. At nine hundred and twenty-seven, the list of names is the bill. Daniel put it in his own decision notes as "deferred tools hide schemas but not names", and he's right.



Corn

So the agent reads the phone book every morning before anyone asks it the time.



Herman

Then he did the useful thing, which was go back through his own transcripts. About seventeen hundred sessions, counting which tools actually got called. Fifty tools covered just over three quarters of everything.



Corn

Leaving eight hundred and seventy-odd in the attic.



Herman

An attic he was paying rent on every morning. Widen it to the top hundred and fifty and you're past ninety percent. So the design that fell out of that scan was a small native list for the regulars, and one door to everything else.



Corn

I went back through how he got here, by the way. Canopy itself is young. Eleven tagged releases in eight days, and one point ten only landed on the fourteenth. But it's grown out of seven months of a private gateway, a skills database that didn't work, and a documentation repo.



Herman

You read the post-mortem.



Corn

I read the post-mortem. The private gateway started in April, on top of MCP Jungle, with tools on one side and skills on the other. The skills side ran on a product called OpenViking, and the list of what went wrong with it is the real design document for Canopy. It was a general context database being used for a small slice of what it could do. It wrote its own summaries with a language model, which duplicated the descriptions Daniel had already written by hand and ran up the bill. Its state lived inside the container instead of in git, so when an auto-updater recreated the container from the latest image with no volume mounted, the whole corpus and the API key went with it.



Herman

That's August. The October one is worse.



Corn

The October one. An image update moved the working directory, and for about three weeks every single call returned unauthorised. The container said it was healthy the whole time. The persistence check was green.



Herman

Green for three weeks.



Corn

There was also a sync walker that marked failed writes as synced. So it was reporting success on things it had never done. He switched it off on the tenth of October, after running the new catalogue alongside it and matching its recall, eighteen out of eighteen.



Herman

You can read every one of those failures straight into Canopy's rules. Failures are loud. Images are pinned. You verify the destination, never the state file that claims the destination is fine. When Canopy writes something to the gateway, it reads it back. That habit only exists because something once lied to him for three weeks.



Corn

Fine. Now the thesis itself. Tools have discovery, skills don't. Herman, do you buy it?



Herman

Mostly. On the tool side it's settled. Anthropic's API has a tool search tool, and it's stable and documented: you flag tools for deferred loading, the model searches them with a regex or a keyword variant and gets a handful back. Their own docs say a typical multi-server setup can eat around fifty-five thousand tokens of definitions, and that tool search usually cuts that by more than eighty-five percent. They even tell you to keep your three to five most used tools loaded and defer the rest, which is Daniel's top-fifty scan in miniature.



Corn

And skills?



Herman

Skills got the middle tier and stopped. We spent an episode on progressive skills discovery a while back, the three tiers, a description, then the body, then the files the body points at, and the way descriptions get truncated. That's all still true. Claude Code loads every skill's description into the session so the model knows what's there, and caps each one at fifteen hundred and thirty-six characters in that listing. The body only loads when the skill fires. There's no search step in front of the descriptions. Then in the episode about agents with too many skills we did the metadata cost, Codex budgeting roughly two percent of context for the list. Everyone has a budget. Nobody has a search.



Corn

The feature request is almost funny. Someone opened an issue on Claude Code back in March asking for exactly this, deferred loading for skills, a SkillSearch tool. Fifty-two skills, five to eight thousand tokens every session. A stale-bot closed it a few weeks later as not



Corn (continued)

planned. No staff reply.



Herman

There's an earlier one that went the same way in February.



Corn

The bot has a strong position on skills.



Herman

Then the protocol side. MCP has a skills extension now, SEP twenty-six forty, and it's final. It gives you a list call, a get call and a skill resource scheme. Discovery is enumeration: here is everything, plus maybe a pointer in the server's instructions. It explicitly lets a big catalogue return a partial list, and tells hosts not to read that as "no skills". What it doesn't have is ranking or search. Canopy serves those calls from its own API and puts search on top.



Corn

My version of the thesis would be narrower. The plumbing for skill discovery exists. Nobody's put a search box in front of it in the client.



Herman

I'd sign that. Which is where the anatomy starts mattering, because Canopy's answer is to put the search box outside the client entirely.



Corn

Go on then. What actually runs?



Herman

A compose stack, one image. Every Python service in it is the same container image started with a different command. The core is a gateway, which is MCP Jungle, pinned to one exact version. A database, which is Postgres seventeen with pgvector. The catalogue, which is the search service. A toolbox. An API service that also runs the scheduler. A one-shot seed job. And a CLI container, Alpine with the GitHub CLI, the AWS CLI, rclone and restic, so skills can drive real command-line tools with the logins living on the server. Then optional ones: a web UI, Garage for S three storage, Ollama plus a job that pulls the model, a Cloudflare tunnel, a Tailscale sidecar. Fourteen services in the compose file all told.



Corn

Fourteen services so the agent can see less.



Herman

I'll defend that later. What the agent sees is one MCP endpoint, a core group the README describes as about ten tools, and the catalogue and the toolbox are always in it. The flow is three verbs. Search returns ranked IDs, skills and plugins first and a short separate list of tools. Read loads one record in full, a skill's whole text or a tool's schema. And the toolbox has a call-tool function that takes a tool name and arguments and calls any upstream tool that isn't natively listed.



Corn

Search, read, call. And who holds the keys when it calls?



Herman

The gateway. Each upstream server is registered with its stored credentials, and the gateway forwards the call. The client holds nothing. That's the part we went through in the episode on designing a home MCP gateway, the transport and the auth, and Canopy is that argument with the receipts attached. Your laptop, your phone, a cloud agent, none of them carry a token for the thing they're calling.



Corn

Who's allowed to change what's registered, then?



Herman

One service. The API is the only thing that holds the gateway's admin token. The CLI, the skills that manage Canopy, the dashboard, they're all clients of that API. Every write goes through it and gets read back to confirm it landed. And there's no Docker socket mounted anywhere, which means Canopy can't update itself. You run the update command on the host. That's a deliberate inconvenience.



Corn

I like a deliberate inconvenience. It's the only kind I trust.



Herman

There's an identity layer under that too. MCP Jungle has two token types, one for MCP clients and one for users and admins, and neither can do the other's job. Daniel confirmed that by testing, and Canopy creates four identities on top of it: the admin, an agent identity, a services identity and a read-only one.



Corn

About six thousand lines of Python for all of that. It isn't much for fourteen services.



Herman

The gateway is somebody else's code, and a lot of the Python is about living with it. This is the part I find genuinely charming. There's a contract suite that runs in CI against the real MCP Jungle image, one test for every upstream behaviour Canopy depends on. And the behaviours are... quirky. A server's tool list freezes at the moment you register it. Registering the same name twice fails, so Canopy deletes and then registers. A registration can report a timeout when it actually worked. A failed re-register used to cost you the working server, so now Canopy saves the config and restores it. And a group update keeps any list you didn't send it, so groups get deleted and rebuilt instead of edited.



Corn

A test suite whose main job is documenting someone else's personality.



Herman

That's not unfair, and the design doc nearly says it. When Daniel decided on the fifteenth to replace MCP Jungle with his own hub, the line in the doc is that most of the contract suite pins its quirks.



Corn

Hold that. I want the search engine first, because that's the bit the earlier episodes never opened up.



Herman

The source of truth is git. The catalogue clones every Claude Code plugin marketplace listed in a config file, parses out the plugins, skills, commands, agents and MCP server declarations, and writes one Postgres row per record: an ID, a kind, a name, a description, the full text, a hash and a vector. Lose the vectors and you've lost an embedding run, not any data. Three rules on top of that. Failures are loud. Listing and reading work even when search is broken. And cost is incremental, so only changed records get re-embedded.



Corn

What happens to a query, start to finish?



Herman

Filter first. Then keyword search with BM twenty-five, done by Tantivy in memory, with English stemming and field weights, so a match on a skill's name counts ten times a match in its body. Then exact vector search on BGE M three embeddings. Then reciprocal-rank fusion of the two lists, vectors weighted three to one. Then an optional hosted rerank, then shaping the reply. Every stage reports itself as degraded if it fails, and the whole thing falls back to keywords.



Corn

The name counts ten times the body. So the title of a skill matters ten times more than the skill.



Herman

For being found, yes. It sounds like a joke until you look at the rerank. On a fifty-query labelled set, fusion alone put the right answer first thirty-four times. Add a hosted Cohere reranker and it went to forty-three.



Corn

Nine more first places. Nothing that good is free.



Herman

Roughly four hundredths of a cent a search, so it's nearly free in money. It costs about half a second, and we'll get to that. The detail I love is that reranking on the descriptions beat reranking on the full skill text. Give the reranker more to read and it does worse. He tried local rerankers as well and none of them was worth shipping.



Corn

So the agent picks from the description and never sees the skill until it's already chosen it. The description is doing all the selling.



Herman

All of it. Embeddings are the same story of picking the boring option on purpose. BGE M three won not because it topped the table. The top four models were within noise of each other, and it's the one that runs both locally on Ollama and hosted for next to nothing. Local and hosted vectors come out practically identical. So he embeds the bulk on the hosted side and the queries locally, about eighty milliseconds against more than a second for the hosted round trip.



Corn

And Tantivy? There was a pure Python keyword index before that.



Herman

Replaced on the seventh, and not for quality. The Python index held the interpreter lock while it rebuilt, so every search slowed by a hundred and fifty to three hundred milliseconds whenever a sync was running. He swapped the engine to get the lock back, and the quality stayed where it was. He also ruled out SQLite full-text search, ParadeDB, LanceDB, Haystack, Meilisearch and a separate search container, for the record.



Corn

A list of rejected search engines longer than most people's list of search engines.



Hilbert

Hilbert's Anteater Trivia. Living xenarthrans are found only in the Americas. I have spent most of my career on the wrong continent, which explains a great deal and excuses nothing.



Corn

I'll pretend nobody said xenarthran.



Herman

Sync, then, because a search index is only as good as its last pull. There's a job every minute that asks GitHub, in batched GraphQL calls, whether any marketplace repo has a new head. Running those batches in parallel took three hundred and fourteen repos from sixteen and a half seconds to under three.



Corn

Every minute. GitHub doesn't mind?



Herman

It uses about eight hundred of the five thousand points it gives you an hour. On top of that there's a full sync hourly, and a gateway refresh that snapshots the tools, about nine hundred and forty of them for Daniel. The fix that came in on the fourteenth was for third-party plugins, which used to wait up to an hour because the quick pass only looked at his own repos. And marketplaces are treated as untrusted input. Owner and repo or an https address only, no climbing out of the folder, symlinks off, and indexing never runs plugin code. The docs add one plain line under that, that skills are still instructions your agent will follow.



Corn

Sanitise the files, then hand the agent a document that says "do whatever this says".



Herman

The index can't fix that, and he doesn't pretend it can. Which gets us to the part that's changed most recently. Corn, what did you make of the habits?



Corn

That's the bit that turns it from a skills index into something personal. Since one point oh eight and one point oh nine there are four new record kinds: habit, SOP, preference and context. They live in a git-backed volume, one markdown file per record, sorted into folders. An agent can write one with a context-write tool, every write is a git commit, it's reindexed immediately and pushed to a remote hourly. So Daniel's own rules about how he works sit in the same search as the skills.



Herman

His own instructions file has shrunk to match. It's directives only now: search the catalogue before a task, read what comes back, save new rules there. Plus a generated block of one-line habit summaries that a plugin keeps in sync.



Corn

The docs name the limit plainly, too. A rule that has to hold in every session, whatever the task, has nothing to search on. You can't discover a rule you don't know to look for. So those stay in the always-loaded file.



Herman

Do you think search-first works at all, then? Would you trust an agent to look before it acts?



Corn

No. And neither does Daniel. There's a line in the design doc: an agent told to search first still often does not. Hence the hook.



Herman

The prompt hook. Every time you send a message, the connector searches on it and injects up to three IDs, but only on a confident match, a cosine similarity of point six or better. On his labelled set it fired on thirty-seven of forty-nine queries.



Corn

Firing isn't the same as being right.



Herman

Twenty-seven of those showed the right record. And on twenty ordinary prompts with nothing to find, it fired once.



Corn

A nudge that mostly keeps its mouth shut, then. I'd take that over a nudge that talks every time.



Herman

He calls point six a starting value, set on a small sample with hand-written negatives. That's the right amount of confidence for it.



Corn

How good is the search when it's asked properly? Not on the example in the README, on his hard set.



Herman

Forty-nine hard queries, each run three times against the live stack. The right record came first nearly nine times in ten, and it was in the top three on all but a handful.



Corn

Better than I do finding my own glasses. What's it cost the agent to ask?



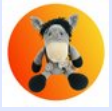
Herman

Median search around five hundred and seventy milliseconds, and search plus reading the winning record comes in around thirteen hundred tokens. Tools on their own are weaker. Their mean reciprocal rank sits about point eight eight, up from point seven four before tools joined the same rerank call.



Corn

Good numbers. One user, though.



Herman

One user, labels drawn from his own catalogue, and a single query moving is worth two points. He writes that down himself. He also notes that two description edits were made with the two missed queries in view, so that part of the gain overstates what a stranger would see. And there are open misses. Tools get crowded out of the vector pool by skills. And a query about where he put his HDMI cable found its skill on the seventh and had lost it by the fourteenth.



Corn

A man with nine hundred tools and no HDMI cable.



Herman

The tools know where everything is except the cable.



Corn

Here's the thing that confused me, and I'd rather settle it than skip it. The README says the agent sees about ten tools. The benchmark has a group called daniel core with sixty-seven tools, at roughly twenty-four thousand tokens. Which is it?



Herman

Both, and they're measuring different things. Ten is the shipped default core group, catalogue and toolbox and a handful more. Sixty-seven is Daniel's personal group, his regulars, the output of that transcript scan.



Corn

And the twenty-four thousand assumes what?



Herman

That every one of those schemas lands in context. It's an estimate, characters divided by four. His architecture notes put the same core at around fourteen hundred tokens, names only, which is what you'd pay if the client defers the schemas.



Corn

So the gap between fourteen hundred and twenty-four thousand is whether Claude Code defers.



Herman

Nobody's measured that yet. The benchmark's own plan says to measure it in a real session before trimming anything. I don't know which of the two his clients actually load day to day, and I'd want that one number before arguing about either.



Corn

Fair. What about where it's going? You said there's a hub.



Herman

Two prototypes, built and tested, and not in the compose file. The shipped stack still runs on MCP Jungle and its groups. The hub replaces MCP Jungle with Canopy's own aggregator. The front door is a single MCP endpoint where the bearer token picks a profile, and it exposes five tools: search, read, list, list servers and call tool. So the catalogue and the toolbox, two things the agent has to learn, become one surface.



Corn

In what order? Ripping out the floor you're standing on seems tricky.



Herman

Front door becomes a compose service. Then the catalogue gets fed from the hub instead of the gateway. Then upstream OAuth, the API routes and the CLI move onto the hub. Then he deletes the gateway, the toolbox, the MCP Jungle database, the compatibility file, the version bump script and the workflow that watches for new MCP Jungle releases.



Corn

That's a satisfying list of things to delete.



Herman

Under the hub there's the access model, decided on the fourteenth and fifteenth, also prototype. Four rules. A space is the only container. Every tool and plugin belongs to exactly one space, claimed by name or by a pattern, so all the Home Assistant tools in one



Herman (continued)

go, and anything unclaimed goes to the operator. A profile is a person plus some of their spaces, with one token each. And a profile sees the union of its spaces. No overrides, no deny lists, no inheritance.



Corn

No deny lists is brave.



Herman

It's the bit I like most. Not allowed answers the same as absent, so you can't probe for what you can't see. It's designed for two people at home, each with a small business, plus a household space and a common space for keyless utilities like the time or exchange rates. And accounts are separate from servers: one server definition, several keys, each owned by a space. Two people's email, one email server.



Corn

That's what a household actually needs. Two businesses, one kettle.



Herman

How would you have done it, Corn? You live with five people and a lot of shared accounts.



Corn

Probably worse. I'd have reached for a deny list on day one, regretted it by day three, and ended up where he did. The union-only rule is the one you get to after the other one hurts.



Herman

Then config, which is where the critique starts, so I'll lay it out straight. There are two folders. Agent data is yours: config, your own skills, notes, a settings file, a secrets file and a shell hook for a secrets manager. Data is Canopy's: Postgres, the clones, tokens, backups. On top of that you can point Canopy at a private git repo as an overlay, and it keeps a read-only clone and pulls it on up or apply. Precedence runs settings, then



Herman (continued)

secrets, then the hook. Non-secret settings only feed compose, never the containers, because in one point oh two a port setting leaked in and broke the API's health check. Fixed in one point oh three.



Corn

And Daniel's overlay?



Herman

Fifty-eight upstream servers, twenty groups, seven marketplaces, and a compose override that adds a second toolbox and catalogue for the household. There's a rule in his notes that a Canopy fix only reaches the live stack when the overlay moves its image pin. Backups run on the scheduler, a daily dump, an hourly config copy and a restore test, all through restic to S three, and CI proves a restore against a throwaway Garage.



Corn

Before the critique, the field. Because "skills don't have this" only means something next to what does.



Herman

Tools first. MCP Jungle is the floor Canopy stands on: a registry, one gateway endpoint, tool groups so a client only sees some tools, and an enterprise mode that wants bearer tokens. No skills. Daniel tested its skills list call on that version and got method not found. Docker's MCP Gateway has a dynamic mode where the agent itself gets tools to find a server, add it, configure it, remove it and execute through it, plus an experimental code mode. Session-scoped, searching a catalogue of servers, and the docs don't mention skills. Cloudflare's MCP portals turned on code mode by default in March, so the agent sees a search and an execute instead of every tool. I've seen two different sets of names for those tools in their own materials, so I wouldn't quote them.



Corn

All tools. All progressive. Nothing for skills.



Herman

On that side, yes, and there's more of it. One MCP does lazy loading with three tools, list, schema and invoke. ToolHive has an optimizer. IBM has ContextForge. MetaMCP is archived. And there's a research thread behind all of them, RAG-MCP, ScaleMCP, MCP-Zero and ToolRet, and the ToolRet finding is the one worth remembering: general-purpose retrievers are bad at picking tools.



Corn

Presumably why he bothered with field weights. Now the skill side, where the thesis gets tested.



Herman

Three serious ones. The agentic community's MCP Gateway Registry puts MCP servers, agents and skill files behind semantic and keyword search. It's real, and it's heavy: nginx, FastAPI, an OIDC identity provider like Keycloak or Entra, MongoDB or DocumentDB, an embedding provider, and Kubernetes or ECS or compose. Obot has virtual MCP servers, tool search on large ones, and a skills registry built from git-backed catalogues with a CLI to search and install approved skills. It's meant for Kubernetes in production, and its quick start mounts the Docker socket, which its own docs mark as dev only. Then K-Dense's claude-skills-mcp, which is the closest in spirit: skills only, vector search, progressive. Daniel's design doc says it's no longer maintained. I haven't seen that confirmed anywhere else.



Corn

And the long tail?



Herman

Skillhub, skill-dock, Vercel's skills CLI, openskills, and Google's agent kit, which has search-skills and load-skill as actual functions. Smithery, which his notes say is now part of Arcade. I haven't checked each of those myself; his own comparison page says every link was fetched on the fourteenth. And one thing I can't tell you from what they publish is whether any of them disclose skills in tiers the way Canopy does. They describe search. They don't describe staging.



Corn

Where does that leave him?



Herman

The idea's not unique. The combination is. A skill index over Claude Code marketplaces, a tool aggregator running unmodified upstream, personal notes as searchable records, and clients that hold no credentials, in one container stack that fits in a home. The enterprise ones need an identity provider and a cluster. The light ones do one half.



Corn

Then the gentle critique, and I'll take the first swing. You promised to defend the fourteen services.



Herman

I said I'd defend it later. It's later.



Corn

Go on, defend it.



Herman

Half of it is optional and the other half earns its place. That's my defence.



Corn

Fine. Then my first cut is the one he's already planning, and I'd just do it first. Ship the front door and delete MCP Jungle before anything else. The contract suite exists mostly to pin another project's quirks. The compatibility file lists exactly one tested version. The aggregator is something he doesn't control and has already decided to replace. Every week it stays, he writes another test about someone else's personality.



Herman

I agree, with one hesitation. The hub's design is done and tested, but the moment it ships he owns aggregation. Every upstream quirk that MCP Jungle quietly absorbed is now his. I'd still do it. I'd just budget for the first month being about upstream servers misbehaving.



Corn

That's a better month than the one where you're writing compatibility tests.



Herman

Probably. Second?



Corn

The record kinds. There are ten: skill, plugin, command, agent, MCP server, tool, habit, SOP, preference, context. The last four are the same object, a markdown file with a description you search on, told apart by which folder it's in.



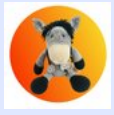
Herman

That's not quite fair. A habit fires before a task's known and an SOP fires on a trigger. The distinction does something.



Corn

It does something for a human browsing. For the search, it's one more filter. I'd have one kind, call it a note, with a type field. Or two, if he wants it: skills, and standing instructions.



Herman

The type field gets him the same filtering with half the vocabulary. Fine, I'll give you that.



Corn

Third one's my favourite. Canopy ships three plugins of its own, a connector, a habits plugin and a manager. Twenty-eight skills between them, to operate a tool that exists so you don't have to load skills.



Herman

Hm.



Corn

That's all I've got. Hm.



Herman

No, it's a real point. There's an easy version, though. Most of the manager plugin is operations, and operations belong in the CLI's help text. The rest could live in the catalogue like everything else, found when you search for "add a marketplace" instead of loaded up front. His private deployment already does that for one of its helper plugins.



Corn

Eat your own cooking, then. Fourth, config. Settings file, secrets file, shell hook, the public repo, a private overlay repo with its own compose override, the context volume with its own git remote, and an image pin you move by hand. That's six places a setting can come from, plus the one decision about when any of it takes effect.



Herman

Every one of those has a reason.



Corn

Every one of them has a reason, and none of them is the reason you're looking at when something's wrong. I'd collapse it to one config file and secrets from the hook only. The overlay can stay, but as the place that one file lives.



Herman

Then my turn. The default footprint. Out of the box you get seven containers, eight with the default search profile pulling Ollama, and Garage storage is on by default too. His own design doc states the goal as one process to secure and one port to expose. The front door is most of the way there, but I'd put the minimal version in writing so the target is visible.



Corn

Which looks like what?



Herman

One container. Tantivy, which he already has, or SQLite full-text if he wants fewer moving parts. One embedding endpoint, hosted or local. Two tools for the agent, find and get. A scheduled job that clones the marketplaces. Skills only, at first. Then the aggregator as an add-on for the people who need tools behind it too. And the design already has the rule that makes this possible: listing and reading work without search. So there's a git-only fallback hiding in there already.



Corn

Two tools. That's the pitch I'd lead the README with. And I'd keep the keyword path front and centre. Thirteen milliseconds median for keywords alone. A hundred and fifty with fused local vectors. Five hundred and seventy once you add the hosted rerank.



Herman

Only you would make BM twenty-five sound like the heritage option you're protecting from the developers.



Corn

It's quick, it runs on the box, and it doesn't send your queries anywhere. I'm not being sentimental. The rerank is off by default for that reason, the queries and descriptions go to a hosted provider. What I'd want is the trade made per query. Nine points of first place for four hundred milliseconds is worth it when the fused scores are close and wasted when the top hit is miles ahead.



Herman

So a confidence gate on the rerank, the same idea as the prompt hook. That's a good idea, and it's cheap to try on the benchmark he already has.



Corn

Last one. The break-even. That SkillSearch issue put fifty-two skills at five to eight thousand tokens every session. Canopy's find-and-load is about thirteen hundred per lookup. So it wins big when the library is big, and on a small library it's probably a loss. Nobody's published where the line is.



Herman

I'd add a second measurement next to that one. All the numbers so far are about search finding the right record. The plan has an item for a real agent doing a real task, with and without Canopy, and it isn't done. That's the one that tells you whether the agent searches, reads and then does the job better. Search scores won't tell you that.



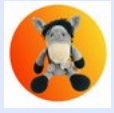
Corn

The search is better than the descriptions it searches. The field weights, the fusion, the rerank, they're all trying to find a record from a sentence someone wrote once, in a hurry, for themselves. Whoever writes the description decides what gets found. Not the rerank, not the vectors.



Hilbert

You're right about that. The council records office put a search terminal on the front counter this autumn. On Wednesday I needed the drawings for the drain under my kitchen. Mrs. Delia Hart typed drain. Nothing. Kitchen. Nothing. The stylus they chain to the counter is not made for claws, so she did all the typing, and she typed for forty minutes.



Herman

What had it been filed as?



Hilbert

Foul water, rear, sundry. Somebody wrote that once and every search since has had to guess it. Your reranking I'm not interested in. I'm interested in whoever wrote sundry. Mrs. Hart got it in the end by going into the back and pulling the green folders, because she knows green is drainage. Herman, you lean into the mic every time you say rerank, come back about an inch… and the terminal does not know green is drainage. When I hand anything to an office, I write on the front, in pencil, in capitals, what I am going to call it when I come back. Drain under kitchen. I did it on the original drawings. Those went to the man who bought my last flat, with the boiler manual, and whoever filed the council's copy never saw my pencil. Corn said earlier that the agent only ever reads the description. I had always thought the pencil was for me.



Corn

Your pencil is the only description in that building written by the person who's going to search for it.



Herman

Which is what that context-write tool is for, now I think about it. The person who'll come looking writes the label. And Mrs. Hart's terminal has the other half sitting in it. Every "drain" that came back with nothing is the word somebody actually used. Canopy logs every query and every result already, and nothing reads that log yet. That log is forty minutes of Mrs. Hart, written down.



Hilbert

It would be, yes. Pencil, though. At the shipping insurer we kept labels in pencil because whoever came after you always had a better word for it.



Herman

That's where I'd send Daniel next. A search log full of the words people typed and got nothing back for is the best description editor he could have. Rewrite the labels from what got searched, in pencil, and let the next person rub them out.



Corn

Thanks as ever to our producer Hilbert Flumingtop, who keeps the levels sane while we argue about container counts. This has been My Weird Prompts. If you want the back catalogue, it's all at my weird prompts dot com.



Herman

We'll be back soon.