Episode Book

EPISODE #5950

Self-Hosting Your Search Stack: SearXNG, Crawl4AI, and the Real Bill

PUBLISHED	14 October 2026 at 08:44 UTC
DURATION	28:42
LISTEN	myweirdprompts.com/episode/self-hosting-search-stack-searxng-50287/
HOSTS	Corn, Herman, Hilbert
SCRIPT MODEL	deepseek-flash
VOICES	chatterbox-regular (chatterbox-local)
PIPELINE	V5.4
SOURCES	11 cited by the research (listed at the end)
BOOK GENERATED	19 October 2026 at 17:41 UTC

SYNOPSIS

Daniel wants to know if you can bolt together a search-and-grounding stack from open-source parts and never rent a search API again. On the table: SearXNG for discovery, Crawl4AI for extraction, and a TypeScript aggregation library for grounding — plus how you'd actually run the whole thing on Modal. The honest answer is yes, with one large asterisk: the bill you save is not the bill you pay. You stop paying Exa and start paying a server, a cron job, and about fifteen hours of your own time. This episode walks through all three layers, the one-line settings change that trips up every self-hoster, the reciprocal rank fusion trick that turns six mediocre result lists into one decent one, and the latency numbers that undercut the whole purity argument.

TRANSCRIPT



Corn

There's a pitch going around that you can now bolt together a search-and-grounding stack out of open-source parts and never rent an API again. Daniel wants to know if that's true, so he's put three projects on the table — SearXNG, Crawl4AI, and something called Agent Web Search — and asked whether they can approximate what a commercial service like Exa does for a deep research agent, how you'd actually run them on Modal, what the real trade-offs are in relevance and latency and cost, and whether the ecosystem is mature enough to drop commercial search APIs from production.



Herman

And the honest answer is yes, with a large asterisk.



Corn

The asterisk being that the bill you save is not the bill you pay.



Herman

Exactly that. You stop paying Exa and you start paying a server, a cron job, and about fifteen hours of your own time you'll never get back. Which is a real trade, just not the one the blog posts advertise.



Corn

So the question underneath Daniel's question isn't can I save money on search. It's what do I take ownership of, and what do I take on as a liability, when I stop renting retrieval.



Herman

Right, and that framing matters because retrieval isn't one thing. A commercial AI search API is doing three jobs at once, and if you're going to self-host you have to reproduce all three.



Corn

Name them.



Herman

Discovery — finding candidate URLs. Extraction — turning those pages into text an LLM can actually read. And grounding — the citations, the dedup, the ranking. Exa sells you all three in one call. The open-source stack sells you three separate tools and a weekend of wiring.



Corn

And the wiring is where people get surprised.



Herman

Always. So let's take the three pieces one at a time, because they sit at different layers and that's the thing that trips people up. SearXNG is a metasearch aggregator. It does not crawl. It takes your query, fans it out to seventy-plus upstream engines — Google, Bing, DuckDuckGo, Brave, Mojeek, Marginalia, and a pile of verticals — and merges what comes back.



Corn

So it's a proxy with good manners.



Herman

It's a proxy with good manners and an AGPL license. It's a fork of searx, started mid-2021, maintained by Markus Heiser. And here's the irony that runs through this whole episode — SearXNG depends on the proprietary indexes it's supposed to be replacing. You aren't escaping Google. You're renting Google through a middleman who doesn't charge you and doesn't tell Google your name.



Corn

That's the whole pitch, isn't it. Privacy, not independence.



Herman

That's exactly what it is. And the API is straightforward once you know the trap. You hit GET slash search with q, categories, language, pageno, time range, and format. Format can be json, csv, or rss. Except JSON is disabled by default, and if you don't know that you get a 403 Forbidden and you spend an hour convinced your container is broken.



Corn

What's the fix?



Herman

You add json under formats in settings dot yaml. One line. Every self-hoster trips on it. Sharma, who wrote up his Docker setup, hit exactly this. And then there's the detail that undercuts the whole purity argument — SearXNG ships an Exa engine adapter. There's a file for it in the engines directory, right next to the Brave adapter, the Jina adapter, the Kagi adapter. So you can absolutely run your pure open-source metasearch engine and have it call Exa on the back end.



Corn

That's not a workaround. That's a confession.



Herman

It's the project being honest about what people actually do with it. And it's worth sitting with, because it tells you the maintainers don't believe the pure version is sufficient either.



Corn

Okay, layer two.



Herman

Crawl4AI. Apache-2.0, Python, eighty-one thousand stars, latest release was late September. It's a crawler built specifically to produce LLM-ready Markdown — that's the entire point of it. Async browser pool, so it's running multiple headless Chromium instances at once. And then the piece that actually matters for RAG, which is Fit Markdown.



Corn

Fit Markdown sounds like a fitness app.



Herman

It's a set of filters that strip a page down to the part worth feeding a model. PruningContentFilterLXML, BM25ContentFilter, LLMContentFilter. You point one at a page and it throws away the nav bars, the cookie banners, the related-articles sidebar, and hands you the body text.



Corn

That's most of the value right there. Anyone who's fed raw HTML to a model knows how much of it is furniture.



Herman

It's the difference between a usable pipeline and a pipeline that burns your entire context window on a newsletter signup box. And then the extraction layer — CSS, XPath, Regex if you want it deterministic, or LLMExtractionStrategy if you want a model to pull structured fields out. Deep crawl with BFS, DFS, best-first, crash recovery. An AdaptiveCrawler. An AsyncUrlSeeder that'll pull from sitemaps or Common Crawl. Stealth mode. Screenshots and PDFs. And a Docker server with a REST API and an MCP interface.



Corn

And the origin story is the tell.



Herman

The author wrote it in the README. He tried the quote-unquote open-source option, it wanted an account, an API token, and sixteen dollars, and it still under-delivered. His words — I went turbo anger mode, built Crawl4AI in days.



Corn

Sixteen dollars. That's the most expensive sixteen dollars in open source.



Herman

It's the emotional engine of this entire movement. That's the thing people miss when they analyze self-hosting as a rational economic decision. Half of it isn't economic. It's somebody getting annoyed enough at a paywall to go build the thing themselves.



Corn

Twenty years of open source has been powered by exactly that.



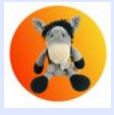
Herman

Layer three. Agent Web Search, and this is where I need to flag an ambiguity before we go any further.



Corn

Because the name isn't one project.



Herman

The name is not one project. There are several repos floating around with near-identical names, and most of them are SearXNG-based backends — basically thin search servers. The most substantial thing matching that name is different in kind. It's hbmartin slash agent-web-search, MIT licensed, by Harold Martin, and it's a client-first TypeScript aggregation library.



Corn

So not a search engine at all.



Herman

Not a search engine. It's the layer that sits above search engines. You give it one query, it fans that query out to many providers in parallel, and it normalizes everything into a common SearchResult and Answer interface. Per-engine isolation, so one provider failing never rejects the whole batch.



Corn

What's it talking to?



Herman

Seventeen engines. Self-hosted SearXNG is one of them. Exa is one of them. Brave, Tavily, Firecrawl, Jina, Kagi, Linkup, Parallel, Perplexity Sonar, SerpAPI, Serper, You.com — plus keyless options like DuckDuckGo, GDELT, and Hacker News.



Corn

So the tool that's supposed to help you avoid commercial search APIs ships with adapters for eleven of them.



Herman

And that's not a flaw. That's the design. It's a routing layer. The reason it's interesting isn't purity, it's that it does the two jobs that are hard. It dedupes results by canonical URL across all the providers, and then it fuses the rankings using reciprocal rank fusion.



Corn

Explain that, because I think it's the least understood piece here.



Herman

Each engine contributes a score to each result of weight divided by k plus rank. Default k is sixty. So a result that ranks first on one engine and nowhere else gets a moderate score. A result that ranks third on four different engines gets a much higher one. You're not trusting any single engine's judgment — you're looking for agreement.



Corn

Consensus ranking.



Herman

Consensus ranking, and it's the single technique that turns many mediocre result lists into one decent one. You don't need a great search index if you have six okay ones and a good fusion rule.



Corn

Which is a different strategy from Exa's, right? Exa is one index, deeply understood, with its own relevance model. This is a committee.



Herman

It's a committee, and committees have failure modes that single experts don't. But they're also harder to buy off — if Exa's index has a blind spot on your niche, you're stuck with it. If one engine in a committee has a blind spot, the others cover it.



Corn

What else does it do?



Herman

formatForLLM — renders the fused results as citation-friendly Markdown or XML, which is the grounding layer, that's what your agent actually reads. Built-in tool definitions for Anthropic, OpenAI, and the Vercel AI SDK, so you can hand it straight to a model as a callable tool. An MCP stdio server. Execution strategies — all, race, fallback, hedged — so



Herman (continued)

you can decide whether you want every provider queried or just the first two to respond. Per-engine throttling, rate-limit awareness, and a client-wide cost budget so one runaway agent doesn't drain your account. It needs Node twenty-two or newer, and it needs a self-hosted SearXNG with JSON enabled.



Corn

Which leads us back to the one-line settings change.



Herman

Everything in this stack eventually comes back to that one line.



Corn

So we've got the three layers. Discovery, extraction, grounding. Now the question Daniel actually asked — what happens when you put this on Modal and point a research agent at it.



Herman

Modal's billing model is well-suited to this. Per second. CPU at about one point three one times ten to the minus five dollars per core per second, memory at two point two two times ten to the minus six per gigabyte per second, egress at four cents a gigabyte. Containers spin up and down in one to two seconds. Default keep-alive is sixty seconds after the last input. And the starter plan includes thirty dollars a month of free compute.



Corn

So for a bursty research task — an agent that does four hundred searches in twenty minutes and then nothing for six hours — you're paying for the twenty minutes.



Herman

You're paying for the twenty minutes, and that's the right shape for this workload. Modal's own webscraper example shows the exact pattern you'd use. Debian slim base image with Playwright and Chromium installed at build time, an async function decorated as an app function, and then you map over your URL list to fan out hundreds of fetches in parallel.



Corn

That's the Crawl4AI pattern almost one to one.



Herman

It really is. Modal built the demo for a different reason but it drops straight onto this stack. And there's a caveat that erases the advantage if you're not careful. Modal's Sandbox product is forced into non-preemptible pricing, which is three times base, and region pinning adds somewhere between one point one five and one point seven five times on top of that. So real sandbox costs run three to five times the advertised rate.



Corn

Which is the kind of thing you discover on the invoice.



Herman

The kind of thing you discover on the invoice in month two, after you've already told your team you're saving money. If your agent runs inside a Sandbox, the per-second billing story is much worse than the headline number suggests.



Corn

Let's talk about latency, because I think that's where the self-hosting argument takes its first real hit.



Herman

Blck Alpaca measured it in production. SearXNG queries took about three seconds. SerpAPI, the thing they were replacing, was two hundred milliseconds.



Corn

Fifteen times slower.



Herman

Fifteen times slower, and the reason is structural. SearXNG isn't slow because it's badly built. It's slow because it queries multiple upstream engines in parallel and then waits for all of them before it merges. The merge is the feature. The wait is the tax that comes with it. They got it below a second by adding aggressive Redis caching and cutting their upstream list down — but it never got to two hundred milliseconds. It can't. You're waiting on other people's servers now.



Corn

And in a deep research agent doing four hundred searches, three seconds each is twenty minutes of pure waiting.



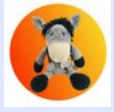
Herman

Twenty minutes of pure waiting that you're also paying for, per second, on Modal. So latency and cost are coupled here in a way people don't always notice.



Corn

Is there a worse breaking point than slowness? Because slowness you can plan around.



Herman

There is, and it's the centerpiece of this episode for me. SearXNG suspends engines after errors. It's a built-in behavior — if an engine throws a rate-limit or a CAPTCHA, SearXNG stops querying it for a cooldown period. One hour for too many requests. One day for a CAPTCHA. Fifteen days for a Cloudflare CAPTCHA.



Corn

Fifteen days.



Herman

And Google, Bing, and Brave fingerprint datacenter IPs, which is what your VPS is. So a self-hosted instance collects suspensions steadily, just by existing. It's not a bug, it's an arms race you're losing slowly. And here's the part that matters — when every enabled engine is suspended, SearXNG does not error.



Corn

It doesn't error.



Herman

It returns HTTP 200 with an empty results array. A healthy-looking response. A script sees a two hundred, sees a well-formed JSON body, parses it, finds zero results, and concludes that the thing doesn't exist.



Corn

And a researcher writes that down.



Herman

A researcher writes that down. This is the grounding catastrophe. An agent that silently records "X doesn't exist" because its search backend was on a fifteen-day timeout isn't just wrong, it's confidently wrong, and it's wrong in a direction that's very hard to detect after the fact. Your citations are empty but your report reads fine.



Corn

That's worse than an API that goes down.



Herman

It's much worse. A commercial API failing loudly is a bad afternoon. A self-hosted metasearch engine failing quietly is a corrupted research pipeline you don't find out about for a month.



Corn

How do you catch it?



Herman

Sharma's approach is the one I'd steal. Before you record any negative finding — any "there's no evidence of this" — run a control query with a known-good answer. If the control query returns nothing, your search is broken and you should not trust the negative. He says it caught the empty-200 twice.



Corn

Two catches. That's two corrupted findings that didn't happen.



Herman

And the control query costs you one extra search per negative finding, which for most research agents is nothing. It's the highest-leverage line of code in the whole stack. He also made an index-choice decision that I think is counterintuitive — he keeps Mojeek, Marginalia, Yep, and Right Dao, and drops the big engines.



Corn

Why drop Google?



Herman

Because Mojeek and Marginalia run their own indexes. They don't need residential IPs, they don't fingerprint you, they don't suspend you. His line is — for research, a narrow index that answers beats a wide one that is suspended.



Corn

That's a real insight. An index that's up beats an index that's better.



Herman

And it reverses the usual assumption. Everyone optimizes for coverage. In a self-hosted context you optimize for availability, because a suspended engine contributes literally nothing.



Corn

What about Crawl4AI's operational side? Because I want to know what actually breaks.



Herman

It ships a full Playwright Chromium, so the image is about four gigabytes uncompressed, one point five four compressed for the AMD sixty-four build. That's not a problem on a server, it's a real problem on serverless where image size affects cold start. Blck Alpaca hit memory leaks after a few hundred crawls — Playwright wasn't cleaning up browser instances properly — and their fix was a cron job restarting the container every six hours.



Corn

That's not a fix.



Herman

Their words. Not a fix, it's a workaround. And the memory tuning is fiddly. Docker needs shared memory set to a gigabyte, because the default sixty-four mebibytes is nowhere near enough for Chromium — you'll get crashes that look like anything except a memory problem. The project's compose file asks for a four gig limit. Sharma runs it at twelve hundred mebibytes, about thirty percent of spec, and says it holds. Meanwhile SearXNG itself uses eighty-two mebibytes.



Corn

So the metasearch engine is a rounding error and the browser is the whole problem.



Herman

The browser is always the whole problem. And then the anti-bot reality — Crawl4AI has no built-in Cloudflare bypass, no proxy rotation, no CAPTCHA solving. None.



Corn

So you hit a Cloudflare page and you're done.



Herman

You're done. Blck Alpaca's fallback is manual review — after three failed attempts, a human looks at the URL. Which is the hidden labor nobody puts in the blog post. Self-hosted doesn't mean unattended. It means the parts that commercial providers handle invisibly with infrastructure you can't buy, you now handle with your own eyes.



Corn

Let's do the money, properly.



Herman

Blck Alpaca's server — SearXNG, Crawl4AI, Redis, Playwright — was eighty euros a month. Then roughly fifteen hours of engineering across three months, which at a hundred an hour is fifteen hundred, plus two hundred forty of running cost. Call it seventeen hundred forty euros. Against a four hundred euro a month API bill, which for the same three months is twelve hundred.



Corn

So they paid about fifty percent more.



Herman

About fifty percent more, in the short term. And their verdict is the line I'd put on a poster. Self-hosting is not a cost-saving model, it is a control model. If you only compare the bill, stick with the APIs.



Corn

Which is the plain answer to half of Daniel's question.



Herman

And it's worth comparing against what the commercial side actually charges, because the numbers are less scary than people assume. Exa's search endpoint is seven dollars per thousand requests for up to ten results, plus a dollar per thousand for additional results. Contents is a dollar per thousand pages. Answer is five dollars per thousand. Deep search is twelve to fifteen per thousand. Agent runs are twelve cents to a dollar per run depending on effort. New accounts get twenty dollars of free credit, roughly twenty-eight hundred searches.



Corn

So a research agent doing ten thousand searches a month is seventy dollars.



Herman

Seventy dollars on search alone, yes. Brave's API is five dollars per thousand requests with five dollars free monthly. Firecrawl is free for a thousand credits a month, eighty-three a month for a hundred thousand, five ninety-nine for a million.



Corn

Against eighty euros a month for the server and a weekend of your life.



Herman

And it's not just money. Exa's latency profiles run from about two hundred fifty milliseconds on instant queries up to twelve to forty seconds on deep-reasoning, with Deep Max claiming eleven seconds at ninety-four percent. When you buy Exa, you're buying a relevance model someone spent years training on a proprietary index. Even with perfect fusion, a committee of free engines doesn't have that.



Corn

This is the part I want to push on, because I think there's a category error in how people compare these.



Herman

Go on.



Corn

We've got good numbers on latency and good numbers on cost. We have no benchmark on relevance. Nobody has run SearXNG plus Crawl4AI against Exa on the same task set and scored the results.



Herman

Nobody has. The closest thing to a quantitative comparison anywhere in this literature is two hundred milliseconds versus three seconds. That's latency, not quality. And the absence is telling — if the open-source stack were matching Exa on relevance, someone would have published the chart.



Corn

Because the chart would be the whole argument.



Herman

The chart would be the entire argument and it doesn't exist. Which means the relevance question is open. It might be close. It might not be close at all. Nobody knows, and the people selling you on replacing Exa are not the people who've measured it.



Corn

That's the thing about this whole movement. It's blog-driven, not forum-driven. Hacker News has almost nothing on running SearXNG inside a search agent.



Herman

Almost nothing. Which for a stack this popular is strange. It suggests the enthusiasm is coming from people writing about self-hosting rather than people running it at three in the morning.



Corn

Let's go back to Daniel's last question, because I want to answer it rather than dance around it. Is the ecosystem mature enough to eliminate commercial search APIs from production deep research pipelines?



Herman

For some pipelines, yes. Honestly yes. If your workload is bursty, if you're cost-sensitive, if your data can't leave your infrastructure, and if you can tolerate multi-second latency — a self-hosted stack is viable. The pieces work. The docs exist. The Docker images are real. Someone published a production Kubernetes stack combining SearXNG, Crawl4AI, and MCP at the end of last year. This isn't a science project.



Corn

But.



Herman

But it isn't mature enough for pipelines that need guaranteed grounding, and the reason is the empty-200. Everything else — the slowness, the memory leaks, the image size, the captchas — everything else is an operational annoyance you can engineer around. The empty-200 is a silent correctness failure. There is no alert, no error code, no retry trigger. Your system reports success and produces nothing.



Corn

And the anti-bot arms race isn't winnable from a datacenter IP, ever.



Herman

Ever. You can win individual battles. You cannot win the war without residential infrastructure, which is a commercial product you'd be buying anyway.



Corn

So the strongest argument for keeping a commercial fallback isn't cost or speed.



Herman

It's that a commercial API fails loudly and a self-hosted metasearch engine fails quietly. That's the whole thing in one sentence.



Corn

And then there's the detail that I think is funniest and also saddest about this entire ecosystem.



Herman

Crawl4AI launched a cloud.



Corn

Soft launch, paid tiers, all of it.



Herman

Free tier at ten requests a minute, two concurrent, two thousand credits. Paid tier at a hundred twenty a minute, ten concurrent, from ten dollars. And the cloud adds three endpoints the open-source version doesn't have — search, serp, and answer.



Corn

Which are exactly the three capabilities the entire self-hosting movement exists to avoid renting.



Herman

It's the self-hosting paradox in one product page. The tool that was built because a competitor wanted sixteen dollars and an API token is now selling a search endpoint.



Corn

I don't even think that's hypocrisy. I think it's the inevitable gravity of this space.



Herman

It's the economics of maintenance. Someone has to pay for the infrastructure that makes the quiet failures stop, and the only people who can charge for it are the people who understand it. The open-source project discovers that the hard part was never the code. It was being on call.



Corn

You can see the endpoint from a mile away.



Herman

Everyone can. That's why it happened. I ran this stack on my own machine for a week and it worked beautifully right up until the moment it didn't, and the moment it didn't, I had no idea why. That's the whole experience in one sentence. It's not that it breaks. It's that you find out from the output, not from the system.



Hilbert

You've got the wrong villain.



Hilbert

The empty-200 isn't SearXNG being badly built. It's everybody treating search like a utility when it's a relationship. I've been running my own setup for the newsletters since March, and I keep a log. Paper. A4, ruled, in the drawer under the desk, one line per day — which engines answered, which ones went quiet, when they came back.



Corn

You keep a paper log.



Hilbert

Forty-one pages since March. I've got four machines on it.



Corn

Four.



Hilbert

Four. Two in the flat, one at my sister's in Haifa, one in a cupboard at the shul — the rabbi thinks it's a boiler controller, which is close enough. And Mojeek's been steady the whole time. Marginalia drops out Tuesdays. I don't know why and I've stopped asking.



Herman

Is the log actually useful, or is it just a record?



Hilbert

It's how I decide when to restart the box. If three engines go quiet in the same week, I restart everything and that clears it. If it's one, I leave it and it comes back. I've been wrong twice.



Corn

Twice in seven months. That's not bad.



Hilbert

Both times it was the Haifa one. My sister unplugs it to run the vacuum.



Herman

Which machine is doing the actual work?



Hilbert

The Haifa one, mostly. It's the fastest. That's why the vacuum matters.



Corn

Do you route anything through a paid API at all?



Hilbert

Never. I'd rather know. That's the whole point of the log — if I paid someone, I'd get a number and no drawer.



Herman

That's a real argument. You're buying observability, not results.



Hilbert

I'm buying the drawer. Anyway, that's all I had.



Herman

The thing Hilbert's describing is the whole episode in miniature. The pitfall you own versus the snag you rent.



Corn

And a paper log is the purest possible version of owning it. Forty-one pages of a man writing down which search engines had a bad Tuesday.



Herman

Cheap to run. Expensive to be the person who keeps it up.



Corn

So if the empty-200 is the real risk — and I think it is — what does a dependable grounding system actually look like? Is it a fallback API, a control-query harness, staff on call, all three?



Herman

My honest read is you need at least two of the three, and the cheapest one to build is the control query. That's one extra call per negative finding and it catches the failure that actually corrupts your output. Everything else is monitoring.



Corn

Which is the same conclusion Hilbert reached, he just reached it with a notebook.



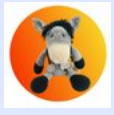
Herman

He reached it with a notebook because the notebook is the monitoring. It's the same instinct — someone has to notice.



Corn

The deeper thing here is that the self-hosting movement was never really about cost. It's about who owns the way it goes wrong. A commercial API fails loudly and bills you. A self-hosted stack fails quietly and you bill yourself, in engineering hours you don't invoice anyone for.



Herman

And the Crawl4AI cloud tells you where this is going. The tools that let you avoid renting an API may themselves become APIs, because the hard part was never the crawling. It was the being awake.



Corn

So the question isn't whether self-hosting works. It clearly works. The question is what kind of failure you can live with — a bill and an error, or silence and a drawer full of log pages.



Herman

That's the one you sit with. Thanks as always to Hilbert Flumingtop for producing.



Corn

If you want more of this, try episode twenty-five oh five, How Self-Hosted Search Actually Works for AI Agents; episode two, Local STT For AMD GPU Owners; and episode seventeen sixty-five, The Agentic Internet. This has been My Weird Prompts. Send us your own prompt on Telegram at t dot me slash MWP listener bot.



Herman

We'll be back soon.



Corn

See you then.

SOURCES

What the research for this episode read before the script was written. Primary sources first.

1. **SearXNG repo (commit 22c64b8, README, AI_POLICY.rst, engine list incl. `exaapi.py`) [primary]**
   https://github.com/searxng/searxng
2. **SearXNG Search API docs (2026.10.13+22c64b8d3) [primary]**
   https://docs.searxng.org/dev/search_api.html
3. **Crawl4AI repo (commit aadd82b, README, v0.9.4, 2026-09-23) [primary]**
   https://github.com/unclecode/crawl4ai
4. **agent-web-search README (MIT) [primary]**
   https://github.com/hbmartin/agent-web-search/blob/main/README.md
5. **Exa API pricing [primary]**
   https://exa.ai/pricing
6. **Modal pricing (2026) [primary]**
   https://modal.com/pricing
7. **Modal webscraper example (Playwright + `.map()`) [primary]**
   https://modal.com/docs/examples/webscraper

8. **Blck Alpaca, 2026-09-04**
https://blckalpaca.at/en/blog/searxng-crawl4ai-why-we-no-longer-rent-apis

9. **Aditya Sharma, 2026-09-02**
https://adityaarsharma.com/crawl4ai-searxng-self-hosted-crawler/

10. **SearXNG + Crawl4AI + MCP Kubernetes stack (2025-12)**
https://github.com/Bionic-AI-Solutions/open-search

11. **Modal sandbox pricing multipliers, 2026-07-09**
https://bex.co/blog/2026/07/09/modal-sandbox-pricing-multipliers