How to build a shitty robot

2026-05-30

"What is my purpose?"

Table of contents

Last Friday I went to the toy store with my boy, and while he was rummaging through the Spider-Man section, my eyes caught sight of a section with very low-cost toy robots.

As I'm playing with agents, LLMs, speech-to-text, and text-to-speech, I thought: why not buy myself one of these low-cost robots, take it apart, and turn it into a fun little LLM-powered toy for my kid and possibly the other kids in the hood?

Or even better: turn it into a STEM learning project that the other parents in the hood and I can do with our kids. That means keeping the work super simple and only using materials that are readily available. There might be some soldering, but I can do that for everybody else.

I went ahead and bought a Silverlit YCOO NEO OCTOBOT (also on Amazon) for 10 euros, which looks like this.

The box set. It has a full LED matrix.

The little robot came with a remote that lets you turn its head counterclockwise or move it forward in the direction it's facing. There was also a little dance button that just randomly turned its head and moved it forward, and some other buttons I didn't even try. The arms are non-functional. You can pose them, but that's pretty much it. Also, that LED matrix is actually not an LED matrix, but more on that later.

First task, disassemble the thing and figure out how it works.

Disassembly using violence

Being the craftsman I am, I naturally didn't have the watchmaker screwdrivers I needed to unscrew the eight-ish or so screws. So instead I used violence.

As it turns out, the advertised LED matrix display was just a few RGB LEDs and a printed inlay on top of it. The inlay is a sort of translucent printed piece with a pixelated rainbow grid and two black eyes on it, with a clear plastic dome sitting in front. The LEDs just shine through it from behind.

With the display removed I had access to the PCB's front side, which showed me a rather trivial layout and set of ICs.

I went ahead and ripped off the top of the robot, leaving me with just the legs, the motor driving them, and the battery bay sporting three triple A batteries. Then I started to trace out the PCB to understand it.

Reverse engineering the PCB and mechanics

The PCB turned out to be really, really simple. I suppose Chinese electronics toys all have this quality to keep costs down. And some of the design was kinda impressive to me as a layman who doesn't have a lot of experience with electronics or mechanical design. Here's the PCB.

In the top left you see the battery connector, which on the back is connected to two capacitors that just make sure the motor gets enough juice in case they stall or draw more power than expected.

The big IC in the middle is the brain. It handles communication with the IR receiver on the middle right, controls the LEDs on either side, and drives the H-bridge, which is the tiny black chip at the bottom.

Interestingly enough, the PCB is a single layer board. If you look closer, you can spot two zero ohm resistors used as little bridges over traces that would otherwise cross.

For me this meant I could just remove the stupid brain IC, which I didn't need, and basically just hijack the H-bridge.

But there was one mystery left. There's only one motor in the robot, but the robot can either walk or turn its head. And the H-bridge only has two of its outputs connected to that motor. So how could the robot turn its head and move forward with just a single motor? Here's how.

The design is rather simple. There are two sets of gears, and depending on the direction the motor is turning, either one of them is engaged. In the back, a tiny gear rotates the top platform, which sets the walking direction via what I can only describe as a transmission. Then if you reverse the motor, the legs move in the direction set by that gear. I found this super genius.

Of course, this also means the robot can only walk forward and can only turn in one direction.

Big plans

Now that I understood how the whole thing works, I started forming a plan. Since I had promised my boy we'd try it out the next morning, I had to move fast. And since I don't have my 3D printer at home, any chassis would have to be built from whatever I had around. That means cardboard.

The next decision was that I'm not going to use my trusty ESP32 boards or one of the dinky little displays I can drive with them, but instead just use my phone as the display and brain of the robot.

With that as a constraint, I had to think of how the phone could control the motor for the rotation and the legs. Luckily enough, I have an Adafruit FT232H at home.

The FT232H can be connected to a phone or computer and gives it kind of the same capabilities that you have on a Raspberry Pi, for example. Specifically, you can use some of its pins as GPIOs, meaning you can send signals into a circuit programmatically. This is exactly what we need to drive the motor driver on the robot's PCB.

That just leaves us with the software. I could have tried to cram everything into the phone with a native app, but having suffered through Android development in the 2010s, I really didn't want to do that.

Instead, I opted for a client-server architecture. The server runs on my laptop and is the actual brain of the robot. It manages speech-to-text, text-to-speech, the LLM, and the agent itself. The ultimate goal here is to have all the machine learning models run locally, so any interactions of my boy with the robot stay private.

The phone is then just a dumb renderer, exposing audio input and output to the server, as well as tools like taking a photo or driving the motor via USB.

Both the server and client side run on TypeScript. On the server I use Node, and the client is just a website served by the server. That in theory makes the client run anywhere there's a browser, except Safari on iOS, because that obviously doesn't implement WebUSB. Because Apple hates the web.

On the UX side, I wanted it to be a friendly, playful, non-sycophantic voice assistant my boy (and other kids in the hood) can talk to and have fun with. It doesn't take a lot to entertain young children, so I didn't want to add too many features. The kids should just be able to speak to it naturally: ask questions, request jokes or stories, have it search the internet, take a photo to read text out loud, play music on Spotify, and obviously walk and turn. I also want the robot to have simple memory, so it can remember past conversations.

I wanted the experience to be as seamless and intuitive as possible, so the speech-to-speech pipeline had to be as good as I can make it within the time constraints of one or two nights.

Rewiring the electronics

This part sounds kinda scary to the uninitiated, but if you have used a soldering iron and a heat gun before, it's actually really easy.

First we need to decapitate the existing robot's PCB. That means getting rid of the big fat black mystery IC. For that I use my trusty heat gun with about 250 degrees Celsius.

It doesn't quite matter if the resistors and caps next to the chip go belly up as well during that process. You just need to make sure that nothing around the H-bridge at the bottom gets fucked up.

Next I soldered two 10cm pieces of wire-wrapping wire to the two top pins of the motor driver.

This can be a little finicky. If you manage to bridge the pins, don't worry. You can use some solder wick, a piece of copper braid you put on top, to get rid of the excess solder. I use a standard soldering iron with leaded solder at 350 degrees Celsius. See my Boxie blog post on the exact gear I use.

Next I soldered those two wires to pins D4 and D5 on the FT232H board. We can then bit-bang those pins via USB to drive the motor. And finally I soldered another piece of wire between the ground pin on the FT232H board and the ground pin on the robot's PCB for common ground.

The mystery IC is still on in this picture. Just ignore it.

And that's all the soldering you need to do. Once that was in place, I had my little coding agent write a single-page HTML file to test the full chain from WebUSB through the FT232H and H-bridge to the motor.

Great success. The wiring was correct and I could drive the motor from WebUSB.

Onwards to the chassis design.

Cardboard engineering

Time to construct the new upper body from cardboard. The first thing I actually did was use double-sided tape to create a sandwich of the battery bay I retained from the original robot, the PCB of the original robot, and the FT232H. So I have a snug little package I can put in whatever chassis I have.

I used some Kapton tape between the FT232H and the original PCB for isolation. You put it between boards to stop them shorting each other. I also used some tape to reel in the wires. But then I got stuck. The round base kinda didn't make sense to me as a chassis, and I couldn't figure out how to mount both the package and the phone on it.

That was until my wife burst in, looked at the thing for one second, and then did this.

I'm obviously married to a genius. It didn't occur to me that I could just put a cup on top of that. She then found a better fitting paper cup, slightly smaller, which I cut a hole into with a craft knife to fit the little package.

I also did a little cutout at the top of the paper cup so the USB port was accessible and through which I could thread the cable from the motor to the original PCB. I then stuck some double-sided tape onto the base of the robot as well as on the lower rim of the paper cup to stick them together.

For my final trick I just had to design the top of this abomination. Since the top rim of the paper cup was nicely horizontal, that was rather easy. I took some cardboard, made some measurements, and then cut to taste.

You can probably derive how this fits together from the images. I used some double-sided tape to secure the fold-ins. Then I put some double-sided tape at the top rim of the paper cup and at the bottom of the top part so those stick together nicely as well. And one more tiny piece of double-sided tape at the front of the top part where the phone sits, so it gets a little stuck there and doesn't fall off. Here's the result.

I'm very happy with this construction. The only part that requires parental help is the craft knife work, but everything else is totally child friendly, including wiring things up. It's also super extensible. The kids can go wild, give it ears, arms, whatever they want.

It's also super easy to repair and super easy to switch out the batteries should they run out.

The boring software

Now that the hardware was in place, I could start working on the software. As I said earlier, it's a client-server design. It's really fucking boring and I don't want to waste a lot of time on it, but here we go.

The general pipeline works like this. The phone continuously streams microphone audio to the server. The server runs voice activity detection to figure out when someone is actually speaking. Once speech is detected, it's transcribed in real time. When the utterance is complete, the transcript goes to the LLM agent, which generates a response and may call tools along the way. Some tools run on the server side, like web search or memory. Others run on the client side, like taking a photo, controlling the motors, or playing music on Spotify. The agent's response is converted to speech and streamed back to the phone. At any point the user can barge in, which cancels the current speech and any running tools, and starts listening again. You can find the server's main orchestration logic in src/server/index.ts.

There are obviously many small details in all of this, but I trust that you and your coding agent can figure that out based on the source code. Here I just want to detail the little journeys I had along the way, mostly involving models, inference engines, and how to get the speech-to-speech pipeline stable enough from a UX perspective.

The first thing I did was find good speech-to-text and text-to-speech models that could run on my M1 Max and serve at least one kid.

Speech to text

For speech-to-text, I already had a lot of experience with Whisper but was never really happy with it. It's essentially a batch model, meaning you give it the full audio and it returns the full text. You can turn Whisper into a kind of fake real-time streaming model, but its performance isn't super great. So instead I tried out Parakeet TDT 0.6B, an int8 quantized ONNX model that runs at 50x real time on my M1 Max. Parakeet is also a batch model, but much faster than Whisper.

I'm using parakeet-rs, an ONNX runtime wrapper to run Parakeet. I wrapped it in a small single-user Rust worker the server communicates with over standard IO, streaming raw PCM audio in and getting JSON events out. The worker handles all the fiddly bits:

- Runs Silero VAD, a tiny neural net, on 32ms audio chunks to detect when someone is speaking
- Once speech is detected, starts buffering chunks into an utterance
- Every 250ms, runs Parakeet on the most recent 4000ms of the buffer and emits an interim transcript. This lets the server detect stop words while the user is still speaking, so it can interrupt the robot mid-sentence if needed (barge-in)
- Once 800ms of silence is detected, runs Parakeet on the full utterance and emits a final transcript

The worker is currently single-user, but turning it into a multi-user worker mostly just means keeping Silero VAD state and the utterance buffer per user. At 50x real time, it should be possible to serve a couple of kids with the same worker.

You might wonder why I didn't just go with the Python version of Parakeet. The answer is that I fucking hate Python. And I'm infinitely sad that the whole machine learning community has decided that Python is the thing we should build production software on. Running it through parakeet-rs and the ONNX runtime is still a bit of a pig, but it allows me to ship mostly self-contained workers somewhere else should the need arise, without having to deal with all the Python badness. uv notwithstanding.

You can just take that worker and use it in your own projects. Easy peasy lemon squeezy.

Text to speech

Initially I got myself an ElevenLabs API key and played around with that. I never planned on using this as the final solution. It's super fucking costly, and I also don't want to send any data to them, even if it's just an LLM's answer to a kid.

I "designed" a little friendly robot voice on ElevenLabs and created a 30-second reference audio file for voice cloning. I then went on a hunt to find a nice open-weights text-to-speech model that can do voice cloning, run at acceptable speeds, and give me output quality that is somewhat close to ElevenLabs. Or at least not absolutely fucking terrible. This is pretty easy for English. For German it's a little bit more complicated. Most text-to-speech models seem to focus on English and CJK.

I played with a bunch of options. Pocketflow TTS is great from a performance perspective, but for German it doesn't quite work. I also tried OmniVoice, which is supposed to be the new king in town, but that too didn't sound anything like German. I tried a bunch of other things but nothing really worked well. Eventually I ended up with Qwen3 TTS, and it's great. Except it's quite a big model comparatively and takes a lot of compute.

The Python MLX implementation of Qwen3 TTS, when using the 6-bit quantized 1.7B base model, runs at around 4x real time on an M5 Max and 2x real time on my M1 Max. That's acceptable for a single user, but as with speech-to-text, I really didn't want a Python dependency.

I ended up with second-state/qwen3_tts_rs, a Rust implementation of Qwen3 TTS based on MLX-C and Rust. It didn't directly work with the MLX model format though, so I started by fixing that. I then found out that the

well-performing Python version achieves its performance by using a 6-bit quantized version of Qwen3 TTS, and that the Rust version had a bunch of bugs that were deal-breakers.

Naturally I vendored the source tree and used my trusty pi with GPT-5.5 to implement feature parity with the Python version. Along the way I found a mysterious bug in the MLX Metal kernels compiled with the latest Xcode. Long story short, I patched it, and now I have a Rust MLX-C based Qwen3 TTS inference engine that gives me the same performance as the Python version without all the Python gunk.

Ideally I would like a cross-platform solution, so that makes me a little sad. But there's only so much time to work on this.

The TTS worker is the inverse of the STT worker. As the LLM streams its response, the server accumulates tokens through a sentence chunker and pushes complete sentences to the worker one at a time via stdin. The worker streams raw PCM audio chunks back out via stdout, both using a simple binary framing protocol. This lets the server start playing audio back to the phone before the full response has been synthesized.

Getting the speech pipeline right

Getting the speech-to-speech pipeline to feel smooth was the trickiest part. A few things worth calling out.

For low-latency responses, the server starts streaming text to the TTS worker as soon as the LLM generates its first complete sentence, using a simple regex-based sentence chunker. It then feeds one sentence at a time, so audio starts playing back on the phone well before the LLM has finished generating the full response.

For barge-in, I first tried WebRTC echo cancellation. The theory is that if we output model speech while someone talks into the mic, the cancellation will remove the model speech, leaving us with just the mic input. But the resulting audio wasn't good enough for STT. So instead, the client runs a custom barge-in detector alongside it. It keeps a ring buffer of playback audio as a reference, and for each mic frame correlates the mic signal against that reference at delays of 20 to 420ms to estimate how much of the mic energy is just speaker bleed.

If the mic RMS is above a threshold AND the unexplained residual is above a threshold, meaning the user is actually speaking and not just picking up the robot, it fires barge-in after 5 consecutive triggered frames. At that point it stops streaming audio to the server and flushes buffered preroll so the server can pick up the utterance from the start. The server then handles stop-word detection via interim transcripts and cancels TTS and any running tools.

LLM and agent

In my quest to run everything locally, I started to get up to speed with the latest local LLM news. Given my M1 Max only has 64 gigabytes of unified RAM, I was looking for a smaller mixture of experts model. I ended up testing Qwen3.6 35B A3B Q5_K_M and Gemma 4 26B A4B Q4_K_M. These are both pretty capable tool callers for their size. They're also multi-modal, which means the kids can show them stuff.

I picked llama.cpp as the inference engine, which worked brilliantly out of the box. I also found that it can easily serve up to four children on my M1 Max with a sizable context window, and quite a bit more on my M5 Max with 128 gigabytes. So that's great.

By default, Pipi uses the Gemma model, just because I found it to be a little nicer from a personality standpoint.

The agent harness is obviously based on pi. I'm using the new abstraction I'm working on as part of the big refactor.

How it started, how it's going

The day after we bought the toy, the boy was super excited. Sadly, I wasn't quite done. I had cobbled together the hardware and a first iteration of the software, but it wasn't ready yet. So instead of showing him the full robot that morning, I just showed him the software running on my laptop. This was still using ElevenLabs for text-to-speech and Claude Haiku as the LLM, and didn't have the full speech-to-speech pipeline yet.

But the test was a great success. The boy loved the interaction with the machine and basically showed it all his toys. We also played a little quiz where the boy would have to guess animals based on descriptions, which proved to me it was worth working on this.

After another night of working on integrating the software with the hardware, we took the little robot outside into the hood for its first field test. Six kids gathered around it and went kinda nuts. The outdoor environment proved kinda hard to handle at that point. Multiple kids speaking across each other isn't something the system can really handle well. They eventually figured out they have to take turns and find a quieter environment. The stupid parents kept babbling too. So they took the little robot into a hut and had it tell stories, jokes, move around, tell the kids what it sees, and so on. It was great fun observing that.

The following nights I kept working on the speech-to-speech pipeline, making it more robust. Here is the final version, or at least what I think is the final version of the bot.

Here's how much RAM this uses on the server.

Still a few minor bugs to fix. The initial codebase was 100% vibe coded. I spent another night refactoring it by hand and with the help of my coding agent, so it's a little easier to extend in the future. I've also added support for Spotify, so the little bot can search, play, and control music or audiobooks for the kids.

A bunch of the kids actually came up to me and asked how to build a robot, so now we have six of those little fuckers.

The next time it's a rainy day we already agreed that we will all gather in our flat together with their parents and each build a robot. It's glorious.

The past months have been mentally exhausting. I'm kind of sick of the entire AI landscape. It all feels pointless. This little project has given me back a bit of my spark. We can still build stuff that delights humans, young and old and in between, using this technology for something worthwhile.

This page respects your privacy by not using cookies or similar technologies and by not collecting any personally identifiable information.

Getting your hands on a capable AI model is the easy part now. Every team can reach the same frontier models through an API, so a strong model is not what sets a product apart. What separates a working product from a demo is everything around the model. You have to measure whether the agent is actually doing its job, then keep grinding on reliability until it stops making expensive mistakes in front of real users.

I moderated a panel on exactly that moderated a panel on exactly that at DigitalOcean's Deploy 2026 conference in San Francisco, a forty-minute conversation with four founders on what they've learned shipping agentic products that people depend on:

- **Angela Hoover**, co-founder and CEO of Andi AI, an ad-free consumer search engine that pairs generative AI with live web data to give people direct answers instead of a page of ad-heavy links.
- **Alex Mashrabov**, co-founder and CEO of Higgsfield AI, a platform that lets creators and agencies produce cinematic video without any physical production.
- **Hovsep Seraydarian**, co-founder and CTO of LawVo, a Canadian legal platform that pairs hundreds of AI agents trained in specific legal areas with human lawyers who verify their accuracy.
- **Peter Elias**, founder of Probably, a data analysis agent that lets non-technical people query their data in plain English and runs calculations on a local engine instead of an LLM so it can decline to answer when the data does not support a clear result.

The discussion got into what each founder underestimated once their agents had to run at scale, how they choose models from a field that keeps growing, what "agentic" actually means in production, and where a real moat comes from when everyone builds on the same foundation.

Watch the full session from Deploy 2026:

Making agents work in production

When the founders were asked what they underestimated once their agents had to run in production, none of them pointed to the model.

You need creative DNA

Higgsfield spent a year on R&D without traction. What finally moved the product was bringing on people who understood how creative work actually happens, then putting them next to the engineers every day.

> "We started to see success when we got non-technical people on the team, like ex-creative directors, who now work daily with engineers to wrap this powerful technology and make it accessible for creatives."
>
> Alex Mashrabov, Higgsfield AI

In high-stakes domains, AI plus humans beats AI alone

LawVo assumed its agents could handle legal guidance with little human involvement. That did not survive contact with real users.

> "We need human lawyers to verify the data and test these agents every day."
>
> Hovsep Seraydarian, LawVo

When asked whether that human role shrinks as the agents get smarter, Hovsep said the opposite is happening. The team watches what its lawyers do and folds that judgment back into the agents one step at a time, which means leaning on people more, not less.

Measurement infrastructure is unavoidable

Peter's point was that one of the first problems you have to solve when you build an agent is the infrastructure that tells you whether the thing works at all. Probably went through several rounds of this until it built an analytics system that watches everything the agent does, and the product now evaluates its own behavior.

> "As long as we record everything the AI is doing, this particular AI can now actually aid us in improving its own performance."
>
> Peter Elias, Probably

Angela made the same point from the oversight side.

> "We're still early on in implementing agents at Andi. We've noticed that when you let them run wild, they'll do anything. You really do have to make sure that they get high quality, accurate, grounded data. We keep an eye on the agents that we have running; we haven't let them be fully autonomous."
>
> Angela Hoover, Andi AI

Model selection is a four-variable problem

The number of available models has exploded over the past year, with frontier releases from Anthropic and OpenAI followed within weeks by open-source alternatives. Capable models are everywhere now. It's challenging to choose among dozens of them when each carries a different mix of cost and capability.

Peter broke the decision into four variables he is always trading against each other: cost, latency, intelligence, and capacity. Smaller models tend to be faster and cheaper but give up intelligence, and an agent that fires many parallel calls runs into capacity limits fast.

> "You want to get to the dumbest model you can get to before you actually go below the product performance that you need."
>
> Peter Elias, Probably

He also warned that users have less patience than founders expect.

> "People will start to get impatient. We found that users were more latency-sensitive than we wanted them to be."
>
> Peter Elias, Probably

Alex runs evaluations every week at Higgsfield because the proprietary data about how users take action has to stay current as models change. He has also moved away from fine-tuning small models toward prompting larger ones, which he finds faster with fewer hallucinations.

> "The rules of machine learning do not change. Understand the customer, understand the business goal."
>
> Alex Mashrabov, Higgsfield AI

Hovsep's rule for any new startup is to start on a frontier model but architect for independence, so only a small slice of the system depends on the LLM and the rest lives in your own application and orchestration. Angela took the lean path from day one, relying on open-source models wherever they were good enough at a lower cost.

'Agentic' doesn't mean autonomous

None of these founders treat "agentic" as a synonym for autonomous. I asked what happens as agents move from co-pilots toward systems that act on their own, and what guardrails that calls for.

Hovsep described a legal field run by dinosaurs, where regulation moves slowly and full autonomy is simply not on the table.

> "Regulations won't allow you to go fully autonomous. You literally get shut down if you do that in this space."
>
> Hovsep Seraydarian, LawVo

What makes that constraint interesting is that LawVo's agents already beat human lawyers on accuracy.

> "We have 92% accuracy on our average agent performance. Your average lawyer has 87%. If you go to a lawyer 100 times, 13 times they're going to make a mistake. We're paying for that."
>
> Hovsep Seraydarian, LawVo

Peter pushed on the word "agent" itself.

> "Agency is the ability to spontaneously take action with no external input. LLMs are not agents. They don't have agency. That's why we have to prompt them."
>
> Peter Elias, Probably

His view is that an LLM never acts on its own. You point it in a direction and keep pushing until it produces what you want.

> "It's being poked with a stick in whatever direction you're trying to get it to do something."
>
> Peter Elias, Probably

That has a practical consequence for anyone building. A model cannot reliably check its own work, and stacking one model to verify another tends to break down, so a human stays in the loop to verify. Peter pointed to the experiment where Claude was put in charge of running a store and lost a remarkable amount of money, the kind of failure that shows up the moment you take human judgment out. His read on the fear that AI will replace everyone is that it is overblown, because these systems are not agents in any real sense. We're just calling them that.

Angela put the actual job of an agent in plain terms. Building one means doing the prompting on the customer's behalf. A task that might take fifty prompts by hand gets compressed into a single step, so the person states the outcome they want and the product runs the prompts behind the scenes and hands back the finished result.

The moat is in the execution

Access to foundational models is getting commoditized. Open-source alternatives trail frontier releases by weeks, and any team can build on the same intelligence. When everyone can reach the same models, what actually sets a company apart?

Hovsep had a simple test.

> “There are startups that are science projects, and there are startups solving real-world problems. Are you solving a real-world problem? That’s it. It ends there for me.”
>
> Hovsep Seraydarian, LawVo

Peter described where the value is going right now. Some of it flows to the labs training the models. A lot of it flows to the inference platforms in the middle, which are making enormous money simply by running GPUs. The application layer holds value because getting these products to behave reliably is genuinely hard, and that reliability is the moat.

> “I could race anybody in building an agent, and it would be: how fast until your agent is as reliable as mine? I will probably win that race because I spent two years getting it to not screw up. That is the moat.”
>
> Peter Elias, Probably

Reliability also compounds. A product that works attracts users, the users put their data into it, and that data makes the next version better in a way competitors cannot copy. Peter also pointed to where he thinks the biggest opening is. Software can finally speak plain English, which means whole categories of tools that were stuck with tiny markets can suddenly reach far more people, because the only thing holding them back was an interface too complicated for a normal person to use.

Angela’s own moat is the data underneath Andi. It started as a consumer search engine, and building it surfaced something more valuable, which is data accurate enough for other systems to depend on. That data has turned into a business of its own as more AI agent companies look for a trustworthy source to ground their answers.

> “There’s a lot of AI agent companies now that need access to high quality, accurate, grounded data.”
>
> Angela Hoover, Andi AI

Both the product and that insight came from the same place: the work.

> “When you’re actually in the trenches building, you learn some insightful things, and then you can build out your moat.”
>
> Angela Hoover, Andi AI

I ended by asking each founder for one piece of parting advice, and the through-line was demand. Alex framed it as a warning, that too many AI companies build for other AI companies and never check whether real customers want what they are selling. Angela put it more directly, that you should talk to your users and test their willingness to pay as early as you can. The most capable agent in the world is still just a demo until a customer pays for it.

DigitalOcean’s AI-Native Cloud is built for teams at every stage, from testing early demand to scaling into the enterprise. It’s one integrated stack from silicon to agent runtime. You get more than 70 models on a single

endpoint, with an Inference Router that handles model selection for you. One API and one bill, with economics that improve as you scale.

→ Get started with DigitalOcean’s AI-Native Cloud

About a year ago I was visiting a friend in Pittsburgh who talked about how much he missed his old social circle in NYC. I suggested he invite some of the people he wanted to get to know to dinner, and he said he didn't know where to start. I said it's Tuesday, so obviously we should make tacos.

He was resistant to the idea of throwing a 'dinner party' so last minute. But we put together a simple invite and he sent it to ten people. We stopped by the grocery store to pick up taco ingredients for about the cost of what he would have paid for a single meal at a restaurant.

In the end, eight people showed up and had such a good time that they made a WhatsApp group to share future plans. There are now a regular set of dinners, plans to go to events, and everyday banter in that WhatsApp group - in other words, an entire friend group spawned from one impromptu dinner.

The first of now many dinners at the Carriage House in Pittsburgh

Eating together is one of the best ways to build community. Priya Rose and her husband hosted regular Sunday potlucks to convince their friends to move near them, which seeded the Fractal network in NYC. Chris Murphy gathered people for regular meals to convince them to join his cohousing project in Mexico City.

It's also really good for you. Studies have shown that eating meals with others lowers stress, lengthens life expectancy, decreases obesity, and helps teenagers make fewer stupid choices.

Yet many adults these days have never built a habit of sharing meals. For some it's too expensive to go out, or they're not confident in their cooking abilities. For others it's too much of a hassle to plan something. And for many the biggest hurdle might simply be in their own heads: they imagine they need to do something 'extra' to invite people over. I recently listened to a podcast where Ezra Klein was interviewing Priya Parker of *The Art of Gathering* fame. Klein agonizes over how he **wants** to start doing regular shabbat meals with friends, but can't seem to get over the hump to actually organize something.

To Klein I would say the same thing I said to my friend in Pittsburgh: just invite them already. Gathering people for a meal doesn't have to be fancy or take a ton of work.

To make it easy for you, here are a few ideas for shared meals that we've seen work well.

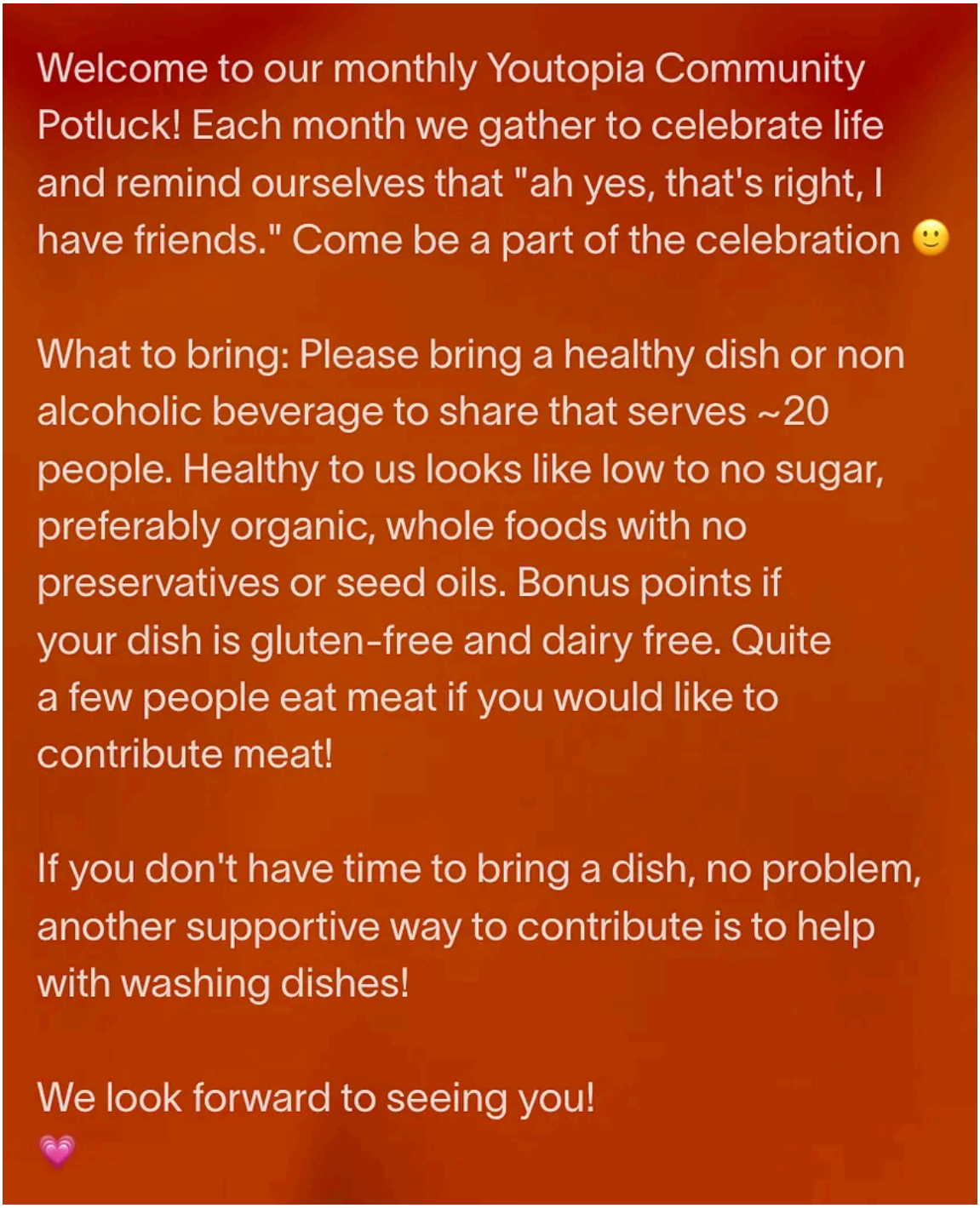
Potlucks: cheap, easy, and delightful

The easiest version I've heard of this is the 'Half-Assed Potluck', which Priya Parker describes in the podcast I linked above:

> The rules are simple: Bring whatever is in your fridge or pick up something on the way. Wear sweats. Don't clean. Use paper plates. Eat what appears, pile onto the couch, talk, laugh. Everyone is home by 8:30.

Potlucks outsource some of the hassle of organizing a meal by asking the attendees to bring some (or all) of the food. As the host, you're providing the venue and giving people a reason to come together. That's a reasonable trade.

The only real 'risk' of potlucks is that you don't end up with a balanced meal if, for example, everyone decides to bring dessert. To mitigate that, you can choose to have people sign up in advance to say what they'll bring - a shared google spreadsheet works well for this - and/or give some gentle guidelines on what kind of food you'd be most excited about:



You can also combine a potluck with another fun event like a clothing swap:

Simone, Raylene, and Elisa are joining forces to host our biggest swap (and brunch) event yet! Come to Michael and Elisa's House with anything and everything you've been meaning to donate, a brunch item in hand, and an appetite!

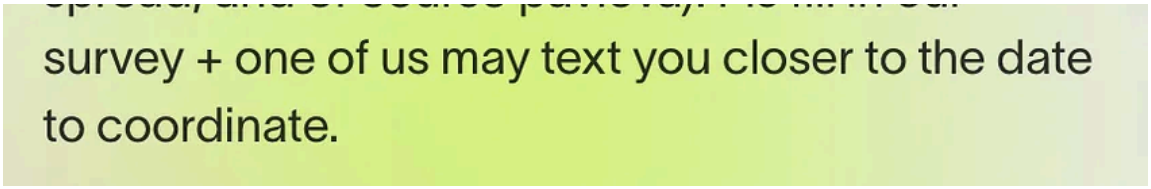
✨✨HOW IT WORKS - PLEASE READ✨✨

THE SWAP:
Marie kondo your house, kitchen, and your closet + bring what no longer sparks joy:

👖clothing, men's or women's (washed, good condition, no rips or stains)
👠shoes, jewelry, accessories
💄 make up (minimally used/ still in good condition)
🍽️ dinnerware / cookware
🖼️ decor (eg vases, candles)
📖 books
❓ anything else!!!

Take home anything you want at the swap!! Everything that gets left over, we'll donate.

THE BRUNCH:
🥥 Please also bring a brunch item to share (cut fruit, pastries, etc) to add to the spread of the dishes / food we'll be providing (eggs, bagels spread, and of course pavlova). Pls fill in our

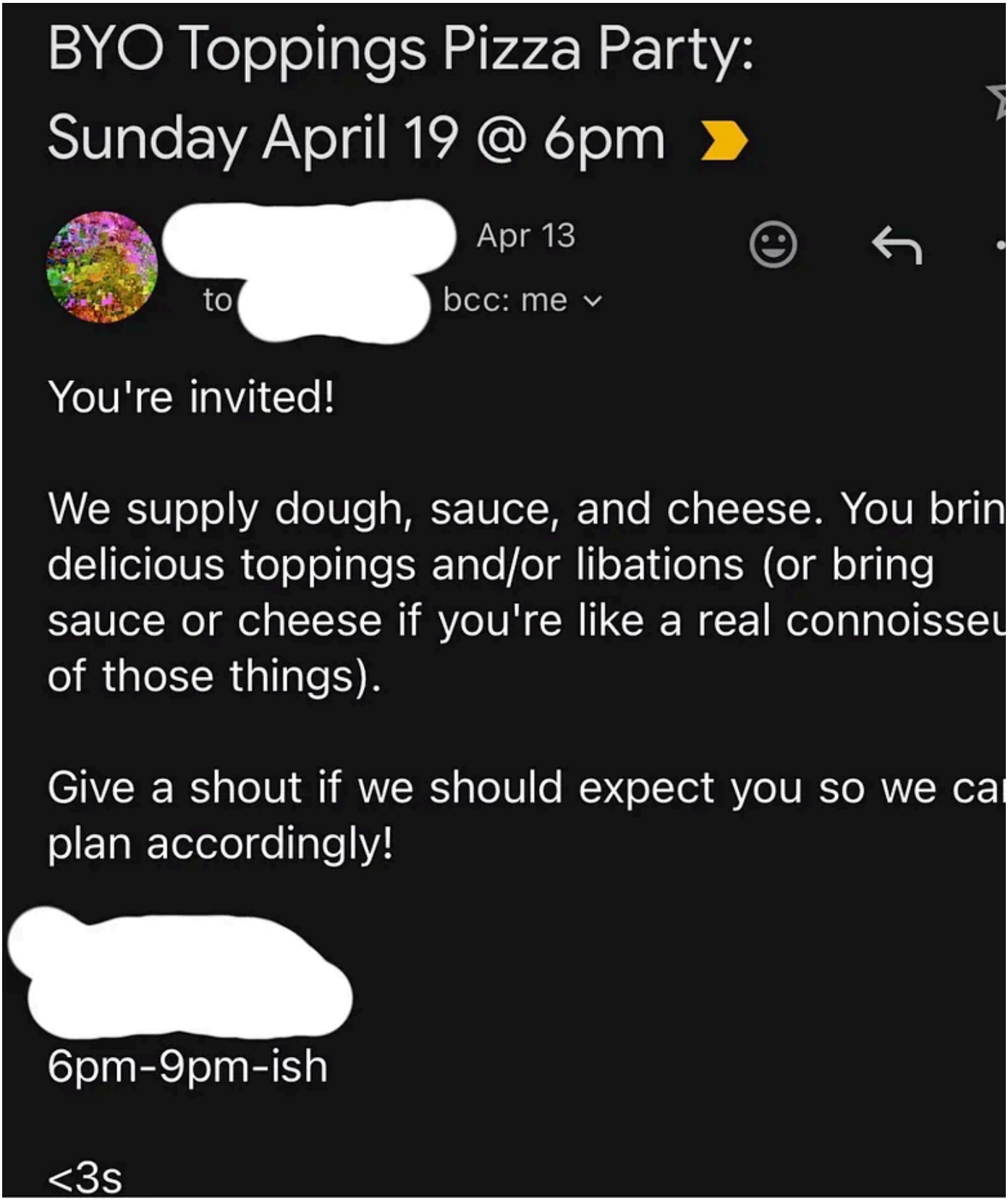


Or a talk about something interesting:

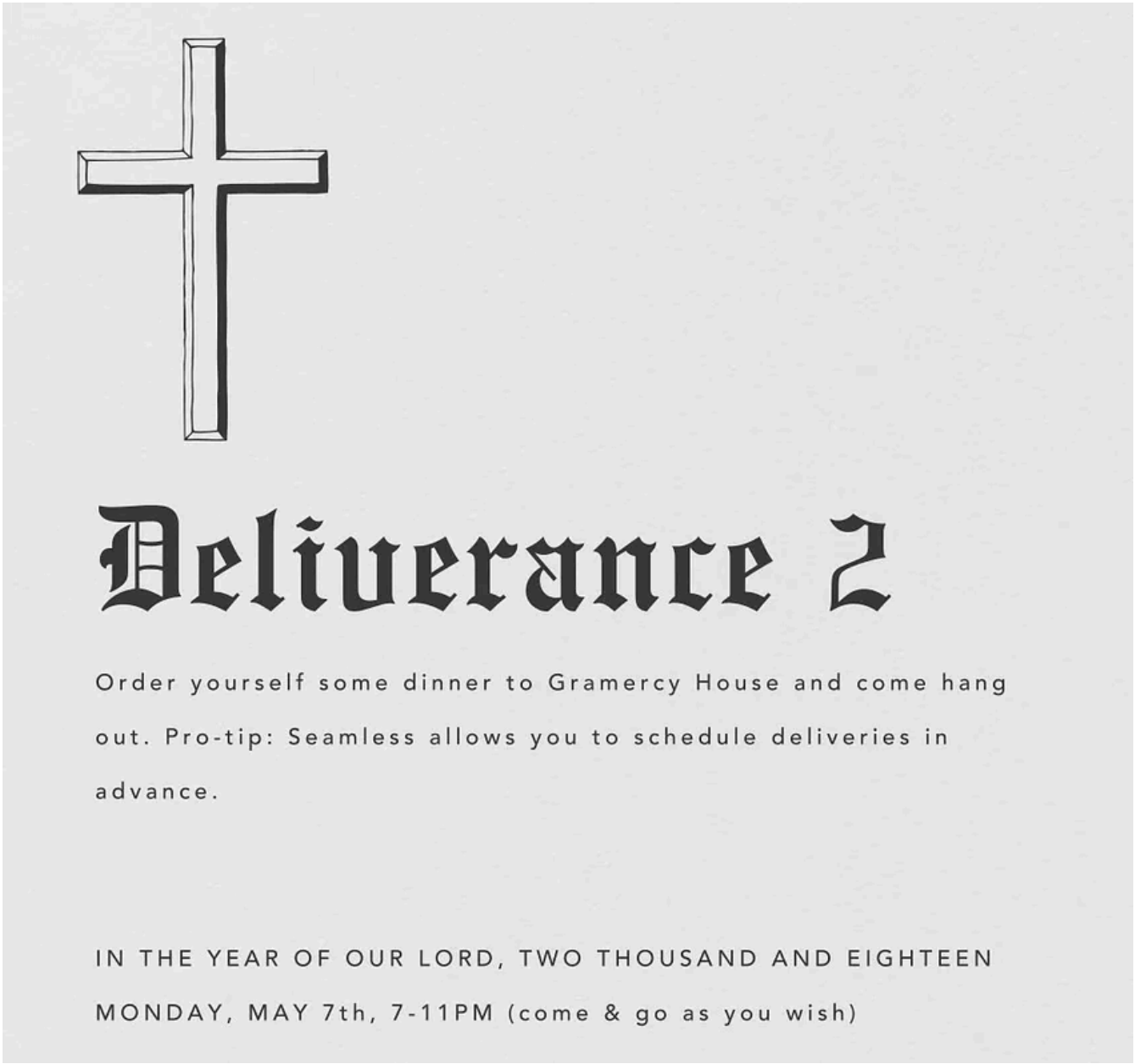
One variation on a potluck I love involves a one time investment in a pizza oven, and buying pizza dough (most bakeries and grocery stores will sell this). You invite your friends to bring pizza toppings - traditional or weird.

It’s fun for your guests to have a thing to do, and you can make exactly as much pizza as people want to eat. My friend swears by this oven which turns out perfect pizza pies in 60 seconds.

The cleanup is minimal and you usually end up with a lot of great ingredients for your home cooked meals going forward.



Deliverance is one step even lower maintenance than a potluck. Instead of asking people to bring a dish, have them simply order delivery to your house at a set time. You end up with a mix of your friend's favorite cuisines, and you can send them home with any leftovers so there's zero cleanup.



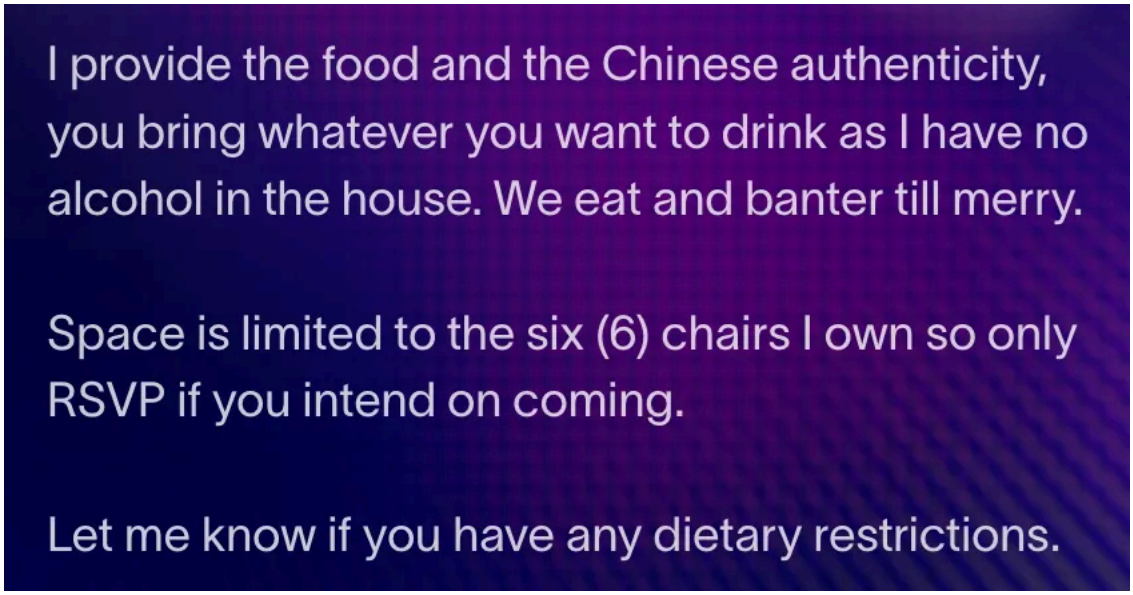
Credit to Kennedy (of the Carriage House in Pittsburgh) for designing this flyer

For any of the above, you don't even technically need to provide a space to host: you can meet at a public park for a picnic.

If you want to provide the food:

'Assemble your own' tacos, salad, or poke: This is a great option if you don't want to think too much about cooking. Another bonus is that it's adaptable to people's dietary restrictions: if they're gluten free or vegan or allergic to nightshades (what are nightshades?) or whatever they can just pick the ingredients that work for them. Phil was the one who taught me about poke - all you have to do is cut up some fish or tofu and then let people assemble the rest, but because it's *poke* people think it's fancy and give you a lot of credit for putting together a great meal.

Share your culture: Here was an invite I got from a friend who loves cooking traditional hot pot:



Don’t make it too hard on yourself

It’s true that hosting takes some energy - you are providing a space that you maintain (unless you’re picnic-ing), and will have to deal with some amount of cleanup. Therefore it’s not unreasonable to ask your guests to contribute in some way. In some of the examples above, the hosts have small concrete asks: help with dishes, contribute $ if you can’t bring food, RSVP only if you intend on coming.

Some people might balk at asking their guests to do anything because they feel that it wouldn’t be good hospitality. To them I’d say: if you make hosting expensive and time consuming for the hosts, they’ll do it less. Most guests are probably more grateful to you for giving them a reason to convene than they are irritated by the request to put their dishes in the dishwasher.

And we need more hosts. People are so hungry for connection they’ll talk to AI chatbots just to feel heard. You have a table. Use it.

To conclude, I’ll outsource to Priya Parker once again:

> Just start. Just start. We’re all sort of sitting there being like: I wish I were invited. Host! One of the most powerful ways to begin to feel like you belong to a place, especially if you’ve moved to a new place, is to host.

One of our countless no-planning potluck dinner parties at Casa Chironja, Puerto Rico

---

Thanks for reading Supernuclear! This post is public so feel free to share it.

Share

Suggested further reading:

Building neighborhood communities

Savannah Kruger
·
September 6, 2024

3 Case Studies on making community with the folks already there

Read full story →

- May 29, 2026

How to redirect if a page doesn't exist with Apache

Over a decade ago, I decided I wanted my article URLs to live at my site's root path `/` instead of a subpath (like `/blog` or `/articles`) "for SEO purposes."

That made sense when I had a few dozen articles and hosted with WordPress.

Today, I have nearly 3,000 articles generated as a flat HTML files with Hugo, and navigating my file system on the server is an unruly mess!

I finally decided to fix this, moving all of my articles to the `/articles` subdirectory.

But I also wanted to sure that all of the many, *many* links to past articles continued to work. That's easy to do when they already live in a subdirectory, and much harder when they're at the root.

I didn't want to go through an individually create redirects for all 2,929 articles. But I also can't just blanket redirect everything in the root to `/articles`, because I have lots of other valid root pages.

Luckily, so `.htaccess` magic came to the rescue!

```
<IfModule mod_rewrite.c>
    # 1. Enable the rewrite engine
    RewriteEngine On

    # 2. If the request is...
    #    - NOT already a /articles page, and
    #    - NOT a real file (-f), or
    #    - NOT a real directory (-d).
    RewriteRule ^(articles)($|/) - [L]
    RewriteCond %{REQUEST_FILENAME} !-f
    RewriteCond %{REQUEST_FILENAME} !-d

    # Redirect to your home page (or another page) first
    RewriteRule ^(.*)$ /articles/$1 [L,R=301]
</IfModule>
```

This redirects all root requests to `/articles/<requested path>`, but…

If the requested root page already exists, it doesn't run. It also doesn't run if the requested URL doesn't already contain `/articles/`, which prevents recursive redirects.

Feel free to steal and modify as needed!

In this newsletter:

- I think Anthropic and OpenAI have found product-market fit
- Claude Opus 4.8: “a modest but tangible improvement”
- Notes on Pope Leo XIV’s encyclical on AI

Plus 5 links and 4 quotations and 1 note and 6 releases and 1 tool

---

I think Anthropic and OpenAI have found product-market fit - 2026-05-27

Anthropic are strongly rumored to be about to have their first profitable quarter. Stories are circulating of companies surprised at how expensive their LLM bills are becoming from usage by their staff. I think this is because OpenAI and Anthropic have both found product-market fit.

- Enterprise customers are now paying API prices
- I think they’ve found product-market fit
- And they’re ramping up
- The AI-failure stories around this are pretty thin
- We also know the labs are spending a lot
- API revenue is becoming less important
- April is a new inflection point

Enterprise customers are now paying API prices

I currently subscribe to the $100/month Max plan from Anthropic and the $100/month Pro plan from OpenAI. If you are a heavy user of coding agents these plans are a fantastic deal. I just ran the ccusage tool on my laptop to get an estimate of how much I would have spent if I were to pay for API tokens in the past 30 days and got:

- $1,199.79 for Anthropic Claude Code
- $980.37 for OpenAI Codex

That’s $2,180.16 worth of tokens for $200 - not bad at all! I’m a moderately heavy user of these tools, but I’m certainly not running agents every hour of the day and night.

I had assumed that companies making extensive use of agents were getting similar discounts. It turns out I *could not have been more wrong* about that.

I haven’t been able to track down the exact date, but at some point in the last six months Anthropic switched their Enterprise plan (originally “Claude seats include enough usage for a typical workday” back in August 2025) to $20/seat/month plus API pricing for usage. This story about the change from The Information is dated Apr 14, 2026, but cites an Anthropic spokesperson claiming that the pricing change occurred in November 2025. Existing customers are finding out about the change as they renew their contracts.

OpenAI made a similar pricing change in April. The Codex rate card (Internet Archive copy) currently says:

> **Note**: On April 2, 2026, we updated Codex pricing to align with API token usage, instead of per-message pricing. This change was applicable to new and existing Plus, Pro, ChatGPT Business and new ChatGPT Enterprise plans.
>
> On April 23, 2026, we made this update for all existing ChatGPT Enterprise plans as well, inclusive of Edu, Health, Gov, and ChatGPT for Teachers.

It’s a little harder to decode as they quote prices in “credits”, but as far as I can tell those credit costs are an exact match for the API token costs listed for those models.

All of which is to say that as of April 2026 the “Enterprise” cost for both OpenAI Codex and Anthropic Claude Code/Cowork is the same as the listed API price.

GPT-5.5 (released April 23rd) is 2x the API price of GPT-5.4. Opus 4.7 (April 16th) is around 1.4x the price of Opus 4.6 when you take their new tokenizer into account.

So April saw both leading model companies release new frontier models with a higher API price, *and* both companies now have measures to lock their enterprise customers (who tend to sign year-long deals) at those API prices, not the previous extreme discounts.

I think they’ve found product-market fit

Why these sudden aggressive moves on pricing? Both Anthropic and OpenAI are planning to IPO, but I suspect there’s a more important factor here: I think they’ve finally found product-market fit, with the coding/general-purpose agent products embodied by Claude Code/Cowork and Codex.

Tools like ChatGPT are wildly popular, but that wild popularity has been difficult to turn into revenue. In February OpenAI boasted more than 900 million weekly active users for ChatGPT, but only 50 million - 5.6% of that - were paying consumer subscribers.

Charging $10-$20/month per user is an OK business, but you'd need 1-2 billion subscribers sticking around for four years to cover $1 trillion in infrastructure.

Companies spending $200+/month/user will get you there a whole lot faster - and as noted above, as a power-user I'm at ~$1,000/month in API costs per vendor already.

Coding agents really did change everything. These are tools which burn *vastly* more tokens, but are also quickly becoming daily drivers for the work carried out by extremely well-compensated professionals. Right now that's still mostly software engineers, but a coding agent is a tool that can automate anything you can do by typing commands into a computer... so they are clearly applicable to a much wider set of skilled knowledge workers.

As I've discussed on this site at length, the models released in November 2025 elevated agents to being genuinely useful. We've had six months to get used to that idea now - it's no wonder companies are beginning to spend real money on this technology.

You could argue that ChatGPT achieved product-market fit when it became the fastest-growing consumer app in history back in February 2023... but it certainly wasn't making any actual money back then. Coding agents plus enterprise pricing marks the point when these companies start making *very* real revenue. Maybe even enough to start covering their costs!

And they're ramping up

As further evidence that enterprise agents represent product-market fit for these companies, consider their open job listings.

OpenAI have 703 open jobs right now, of which I'd categorize 229 (32.6%) as relating to enterprise sales and support - account executives, "Go To Market", "Forward Deployed Engineers" and the like.

Anthropic have 390 open jobs, 105 (26.9%) of which look enterprisey to me.

It's pleasingly ironic that these AI labs have picked a business model with such a heavy demand on human labor - enterprise sales contracts don't close themselves without a whole lot of humans in the mix!

(I ran this analysis by scraping their job sites with Claude Code, then having it use Datasette's JSON API to pipe that data into Datasette Cloud where I used Datasette Agent for the analysis, exported here. Dogfood!)

The AI-failure stories around this are pretty thin

I started digging into this in response to a growing volume of stories claiming that large companies were sounding the alarm because their AI usage costs had grown so large.

The most widely cited of these stories appear quite overblown to me.

The most discussed has been Uber, based on this report where CTO Praveen Neppalli Naga indicated that Uber had "maxed out its full year AI budget just a few months into 2026", mostly thanks to Claude Code.

Given that Claude Code only got *really* good in November it's entirely unsurprising to me that a budget set in 2025 may have failed to predict demand for that tool in 2026!

That Uber story was further fueled by comments made by Uber's COO, Andrew Macdonald, on the Rapid Response podcast. I tracked down the segment and there really isn't much there. Here's what Andrew said:

> But then you sometimes go and talk to your senior engineering leaders and you're saying, OK, how many projects that were on the cutting room floor got moved above the line because of the productivity gains because 25% of our code commits were via Claude Code last quarter?
>
> That link is not there yet, right? I think maybe implicitly there's more that is getting shipped. But it's very hard to draw a line between one of those stats and, OK, now we're actually producing like 25% more useful consumer features, right? And that line is hard to draw.

Somehow this fragment turned into headlines like Uber's COO says it's getting harder to justify the money spent on AI tokenmaxxing, because the market for stories about AI failures remains enormous.

The other popular story around this is Microsoft starts canceling Claude Code licenses, ostensibly to encourage their engineers to dogfood their own Copilot CLI agent instead - but The Verge reporter Tom Warren says "sources tell me the decision is also a financial one", triggered by the June 30th end of Microsoft's financial year.

I think both of these stories support my "product-market fit" hypothesis. The best advice I ever heard on pricing a product was that your customer should *suck air through their teeth* and then say yes. Uber's budget overrun and Microsoft's seat cancellations look like that effect playing out in practice.

We also know the labs are spending a lot

The big AI labs spend billions of dollars on both training and inference. Credible figures are hard to come by, but we did get one huge hint as to the figures involved from, oddly enough, the recent SpaceX S-1:

> [...] in May 2026, we entered into **Cloud Services Agreements with Anthropic PBC** ("Anthropic"), an AI research and development public benefit corporation, with respect to access to **compute capacity across COLOSSUS and COLOSSUS II**. Pursuant to these agreements, the customer **has agreed to pay us $1.25 billion per month** through May 2029 [...]

The Anthropic announcement said that this deal meant they could "increase our usage limits for Claude Code and the Claude API", heavily implying that Colossus is being used for inference, not model training.

Anthropic already have vast amounts of compute from other providers. The fact that they're willing to spend $1.25 billion per month for extra capacity from just *one* of their vendors hints at how big these inference budgets have become.

API revenue is becoming less important

Over the past two years my impression has been that OpenAI made more of their income from subscription revenue while Anthropic made more from their API.

Anthropic's API revenue was historically quite dependent on a small number of large API customers - this VentureBeat story from August 2025 quotes "sources familiar with the matter" suggesting that just Cursor and GitHub Copilot were responsible for $1.2 billion of the company's then-$4 billion revenue.

Today Anthropic are rumored to hit $10.9 billion in the second quarter, potentially even operating at a profit for the first time.

This pivot-to-Enterprise suggests that the labs have realized that the real money lies in cutting out the middlemen. Anthropic's Claude Code directly competes with Cursor and Copilot. No wonder Cursor are investing in their own models!

April is a new inflection point

I've called November 2025 the November inflection point because that was when GPT-5.1 and Opus 4.5, combined with their respective coding agent harnesses, got *good* - good enough that we've spent the last six months adapting to agent systems that can reliably get useful work done.

I think April 2026 is a new inflection point where the revenue implications of this have started to land, to the benefit of the frontier AI labs and with material impacts on the budgets of large companies.

We'll know for sure how real this moment is when the S-1 documents for the upcoming Anthropic and OpenAI IPOs give us some real, audited numbers to get our teeth into.

Claude Opus 4.8: "a modest but tangible improvement" - 2026-05-28

Anthropic shipped Claude Opus 4.8 today. My favourite thing about it is this note in the release announcement:

> Users will find Opus 4.8 to be a modest but tangible improvement on its predecessor. There's still more to be done: we're working on developing and releasing models that provide many of the same capabilities as Opus at a lower cost.

It's so refreshing to see an AI lab honestly describe a release as a minor incremental improvement over the previous model!

Honesty seems to be a theme. Here's my other favorite note from that announcement:

> One of the most prominent improvements in Opus 4.8 is its *honesty*. We train all our models to be honest---for instance, to avoid making claims that they can't support. But a general problem with AI models is that they sometimes jump to conclusions, confidently claiming to have made progress in their work despite the evidence being thin. Early testers report that Opus 4.8 is more likely to flag uncertainties about its work and less likely to make unsupported claims. This is borne out in our evaluations, which show that Opus 4.8 is around four times less likely than its predecessor to allow flaws in code it has written to pass unremarked.

That linked system card includes the following:

> Claude Opus 4.8 had the lowest incorrect-rate of the six models on every benchmark—the most direct measure of factual hallucination. It achieved this mainly by abstaining on questions about which it was uncertain rather than by answering more questions correctly.

Model characteristics

Not much has changed since 4.7.

It's priced the same as Opus 4.5/4.6/4.7 - $5/million input and $25 per million output. "Fast mode" is twice that price, which is a significant reduction from their previous models - fast mode on 4.6/4.7 remains at $30/$150. Note that fast mode is only available to organizations that are part of the research preview, "Contact your account manager to request access".

Both the reliable knowledge cutoff and the training data cutoff are January 2026, the same as for 4.7.

The context window is still 1,000,000 tokens, and the max output is 128,000 tokens.

The What's new in Claude Opus 4.8 document has some of the more interesting details. These caught my eye:

> **Mid-conversation system messages**. Claude Opus 4.8 accepts `role: "system"` messages immediately after a user turn in the `messages` array (subject to placement rules). This lets you append updated instructions later in a long-running conversation without restating the full system prompt, which preserves prompt cache hits on the earlier turns and reduces input cost on agentic loops.

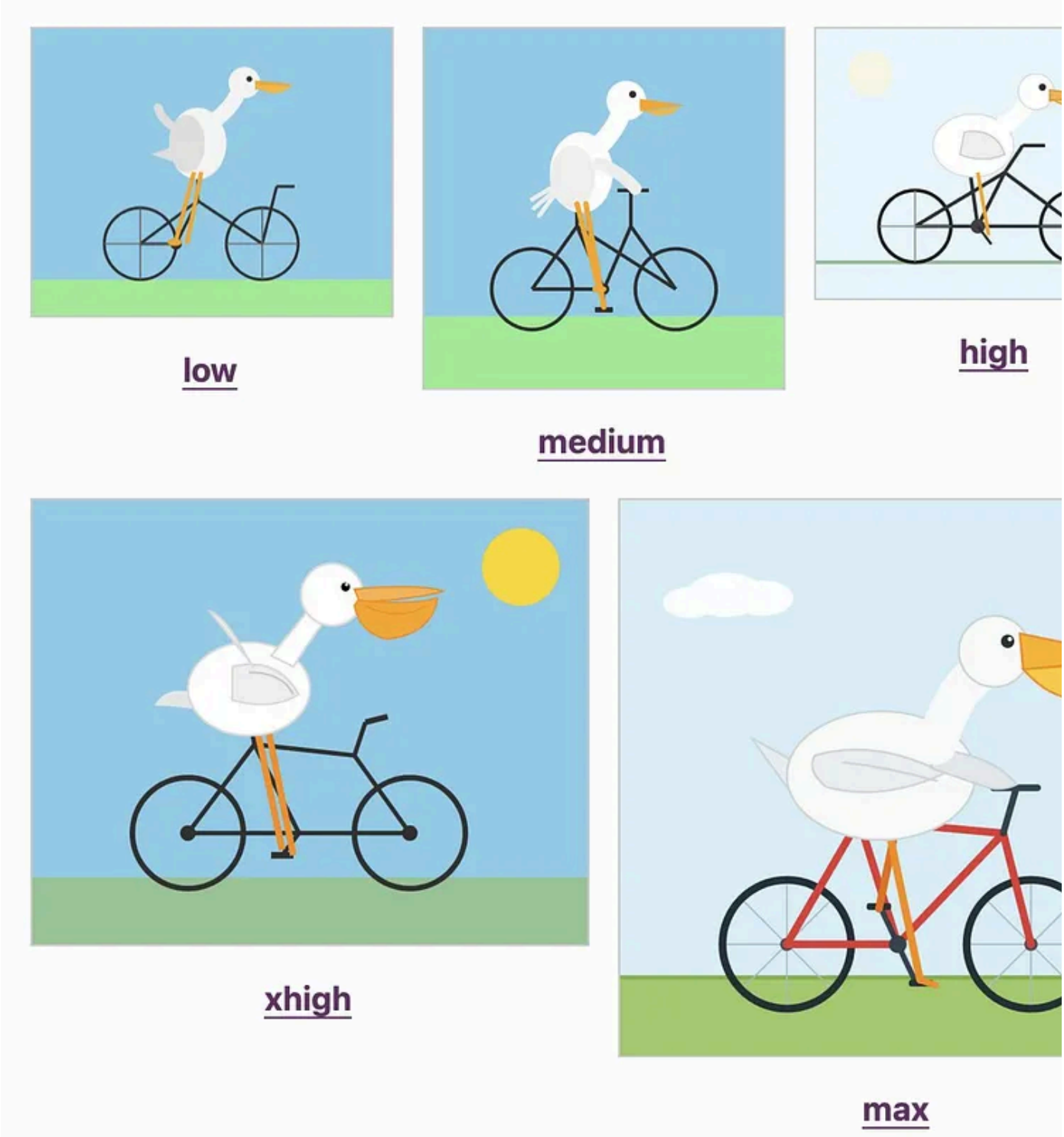
See also this update to the Anthropic Python SDK. Being able to steer the system prompt mid-conversation sounds really powerful. I was worried this would be incompatible with the abstraction provided by my own LLM library, which expects a single system prompt per conversation... but it turns out my recent redesign should handle that just fine.

> **Lower prompt cache minimum**. The minimum cacheable prompt length on Claude Opus 4.8 is 1,024 tokens, lower than on Claude Opus 4.7.

I checked and 4.7's minimum was 4,096.

And some pelicans

Here are pelicans riding bicycles for all five thinking levels, `low`, `medium`, `high`, `xhigh`, and `max`:



This time I ran them using the LLM CLI, exported the logs to Markdown and then had Claude Opus 4.8 build me an HTML tool that could render that Markdown with the `svg` fenced code blocks displayed as SVGs on the page.

(I later had GPT-5.5 xhigh in Codex update that code to remove any XSS holes. I'm sure Claude could have done that if I'd asked, but GPT-5.5 is my code security blanket at the moment.)

The max one was clearly the best, but it did take 25 input, 17,167 output tokens for a total cost of 43 cents!

Notes on Pope Leo XIV's encyclical on AI - 2026-05-25

Dropped this morning by the Vatican: Magnifica Humanitas of His Holiness Pope Leo XIV on Safeguarding the Human Person in the Time of Artificial Intelligence. This is a *very interesting* document. It's some of the clearest writing I've seen on the ethics of integrating AI into modern society.

Pope Leo XIV chose the name Leo in honor of Pope Leo XIII, who is known for his 1891 *Rerum novarum* encyclical on "Rights and Duties of Capital and Labor".

This story on Vatican News further clarifies the significance of that decision:

> Meeting with the College of Cardinals for their first formal encounter after his election, Pope Leo XIV explained part of the reason for the choice of his papal name. "There are different reasons for this," he said, before going on to explain that he chose the name Leo "mainly because Pope Leo XIII, in his historic encyclical *Rerum novarum* addressed the social question in the context of the first great industrial revolution."
>
> "In our own day," he continued, "the Church offers to everyone the treasury of her social teaching in response to another industrial revolution and to developments in the field of artificial intelligence that pose new challenges for the defence of human dignity, justice, and labour."

And now we get Pope Leo XIV's own encyclical on the AI revolution. There's a lot in here, but the writing style is very approachable, including to non-Catholics.

A few of my highlights

(I listened to most of the encyclical on a walk with our dog, my first time trying the ElevenReader iPhone app. It worked very well: I pasted in a URL to the document and it read it to me in a very high quality voice, highlighting each paragraph as it went.)

Here are some of my highlights. In each case below **emphasis** is mine.

Here's a useful description of the interpretability problem for LLMs in section 98:

> First, any statement regarding AI risks becoming quickly outdated, given the remarkable pace at which these systems are developing. Second, all of us, including those who design them, possess only a limited understanding of their actual functioning. Indeed, **current AI systems are more "cultivated" than "built," for developers do not directly design every detail, but instead create a framework within which the intelligence "grows."** As a result, fundamental scientific aspects — such as the internal representations and computational processes of these systems — remain, at present, unknown.

I liked section 83's description of the relationship between development and dignity:

> For individuals as well as for nations, development is both a duty and a right. Minimum conditions are required for enabling every person and people to flourish in accord with their dignity, without being kept in a state of dependence or excluded from access to necessary goods. Development is truly human when it places people at the center instead of the accumulation of wealth, and when it concerns peoples as well as individuals. Justice demands the recognition of the rights of society and the rights of peoples, and includes a responsibility toward future generations. **Development is not truly human if it increases consumption for some while shifting costs and burdens onto others, or relegates entire regions to subordinate roles, preventing them from realizing their full potential**.

Baked in cultural biases and sycophancy get a mention in section 100:

> In personal use, three aspects in particular deserve careful consideration: the ease with which results are obtained, the impression of objectivity and the simulation of human communication. The speed and simplicity with which information, complex analyses, media content and practical assistance can be accessed undoubtedly makes life easier. Yet they can also encourage excessive reliance and the search for ready-made answers, and weaken personal creativity and judgment. **The apparent objectivity of the responses and suggestions these systems provide can lead us to overlook the fact that they reflect the cultural assumptions of those who designed and trained them, with all their strengths and limitations**. The artificial imitation of positive human communication — words of advice, empathy, friendship and even love — can be engaging and at times genuinely helpful. **However, for less discerning users, it can also be misleading, creating the illusion of a relationship with a real personal subject**. When words are simulated, they do not build genuine relationships, but only their appearance. The artificial imitation of care or support can become particularly risky when it enters contexts where real relationships and emotional bonds are lacking.

101 touches on the environmental impact:

> Current AI systems require enormous amounts of energy and water, significantly influencing carbon dioxide emissions, and place heavy demands on natural resources. **As their complexity increases, especially in the case of large language models, the need for computing power and storage capacity grows too, which requires an extensive network of machines, cables, data centers and energy-intensive infrastructure**. For this reason, it is essential to develop more sustainable technological solutions that reduce environmental impact and help protect our common home.

102 covers the risks of algorithmic systems making decisions that impact people's lives without "compassion, mercy, forgiveness":

> The use of AI is never a purely technical matter: **when it enters processes that affect people's lives, it touches on rights, opportunities, status and freedom**. Important and sensitive decisions — concerning employment, credit, access to public services or even a person's reputation — **risk being fully delegated to automated systems that do not know "compassion, mercy, forgiveness, and above all, the hope that people are able to change,"** and can therefore give rise to new forms of exclusion.

105 emphasizes the need for human accountability in how these systems are applied:

> For AI to respect human dignity and truly serve the common good, responsibility must be clearly defined at every stage: **from those who design and develop these systems to those who use them and rely on them for concrete decisions**. In many cases, however, the internal processes leading to a result remain opaque, making it harder to assign responsibility and correct errors. **This is where accountability becomes crucial: the possibility of identifying who must "account" for decisions, justify them, monitor them, and, when necessary, challenge them and remedy any harm caused**.

And 108 touches on the way AI amplifies the power of those with resources:

> In fact, as with every major technological shift, **AI tends to amplify the power of those who already possess economic resources, expertise and access to data**. In light of the common good and the universal destination of goods, this raises serious concerns, since small but highly influential groups can shape information and consumption patterns, influence democratic processes and steer economic dynamics to their own advantage, undermining social

> justice and solidarity among peoples. For this reason, it is essential that the use of AI, especially when it touches on public goods and fundamental rights, be guided by clear criteria and effective oversight, grounded in participation and subsidiarity.

That same section explicitly calls out data as something that should be thought of more as a public good:

> [...] Moreover, **ownership of data cannot be left solely in private hands** but must be appropriately regulated. **Data is the product of many contributors and should not be treated as something to be sold off or entrusted to a select few**. It is necessary to think creatively in order to manage data as a common or shared good, in a spirit of participation, as Saint John Paul II already suggested regarding collective goods.

Given that Palantir is named after a *Lord of the Rings* reference, I can't help but wonder if the J.R.R. Tolkien quote from *The Return of the King* (section 213) was the Pope throwing a little shade at Peter Thiel.

> The twentieth-century Catholic author J.R.R. Tolkien, in the words of a protagonist in one of his novels, described our responsibility in this way: "It is not our part to master all the tides of the world, but to do what is in us for the succour of those years wherein we are set, uprooting the evil in the fields that we know, so that those who live after may have clean earth to till." The civilization of love will not arise from a single or spectacular gesture, but from the sum total of small and steadfast acts of fidelity that serve as a bulwark against dehumanization. For this reason, it is worthwhile pausing to reflect on some aspects of how we, each in our own way, can cooperate in building the civilization of love.

Another 2026 prediction down

On 6th January this year I joined the Oxide and Friends 2026 predictions podcast episode to talk about predictions for 2026, 2029 and 2032. I wrote mine up here, with hindsight they weren't nearly ambitious enough - it's already undeniable that LLMs write good code, we've made huge advances in sandboxing and New Zealand kākāpō have indeed had a truly excellent breeding season.

There's one segment from the episode that I didn't bother to include in my write-up, but that I can't resist providing as a lightly-edited transcript here:

> **Bryan Cantrill:** 37:13
>
> I think that AI has created some real public perception problems for itself. And I think that you are gonna have one of the frontier model companies, this year, have a white paper explaining how the proliferation of AI will mean prosperity for everybody. They will be trying to make some economic argument - because this is gonna be a 2026 election issue, how we think of these things and how they are regulated and it's a big mess. There's more heat than light in this debate.
>
> **Simon Willison:** 38:05
>
> I'd like to tag something on to that one: I think that only works if they can sort of wash that through existing trusted experts. Sam Altman and Dario are constantly publishing essays about this stuff and nobody believes a word they say. Get Barack Obama's signature on one of these position papers and *maybe* you've got something people might start to trust a little bit.
>
> **Adam Leventhal:** 38:27
>
> Otherwise, it's just like "leaded gas is good for you", says Exxon.
>
> **Bryan Cantrill:** 38:31
>
> I mean, yeah. God. Obama... let's go with that, that's a great one because if it's like Bill Clinton everyone's gonna kind of roll their eyes, so it's gotta be someone who's got real credibility saying that this is gonna be broad-based... I'd say if they get that person to do it, it's gonna be revealed that that's also a bit crooked.
>
> **Simon Willison:** 38:57
>
> How about the Pope?
>
> **Bryan Cantrill:** 39:01
>
> The Pope is very into this stuff! That's a great prediction. We've hit pay dirt. The Pope weighing in on LLMs and their economic impact on the world.
>
> Simon, I'm giving you full credit if the Pope weighs in believing that this is gonna be economic devastation.

My prediction here looks a whole lot less insightful given the Leo XIV/Leo XIII relationship, which I was unaware of when we recorded the episode!

Release: datasette-agent 0.1a3

- "View SQL query" buttons for both visible tables and collapsed SQL result tool calls.
- Don't display empty reasoning chunks
- Improved handling of truncated responses - table still displays to the user even if the SQL results were truncated when showing the agent.

See Datasette Agent, an extensible AI assistant for Datasette.

Release: datasette-agent-charts 0.1a2

- "View SQL query" buttons below rendered charts.

Link 2026-05-22 The memory shortage is causing a repricing of consumer electronics:

David Oks provides the clearest explanation I've seen yet of why consumer products that use memory are likely to get significantly more expensive over the next few years.

The short version is that memory manufacturers - of which there are just three remaining large companies - have a fixed capacity in terms of how many wafers they can process at any one time. This fixed wafer capacity is then split between DDR - used in desktops and servers, LPDDR - used in mobile phones and low-energy devices, and HBM - used with GPUs.

Until recently, HBM got just 2% of that wafer allocation. The enormous growth in AI data centers has pushed that up to an expected 20% by the end of 2026, and "a single gigabyte of HBM consumes more than three times the wafer capacity that a gigabyte of DDR or LPDDR does".

Memory companies have learned from the extinction of their rivals that you should always under-provision rather than over-provision your fabricator capacity. The profit margins and demand for HBM (high-bandwidth memory) will constrain the production of consumer-device RAM for several years.

This is already being felt in the sub-$100 smartphone market, which is particularly important to markets like Africa and South Asia.

(The original title of the piece was "AI is killing the cheap smartphone" but I'm using the Hacker News rephrased title, which I think does more justice to the content.)

Link 2026-05-23 On the <dl>:

I learned a few new-to-me things about the `<dl>` element from this article by Ben Meyer:

1. A `<dt>` can be followed by *multiple* `<dd>`
2. You can optionally group the `<dt>` and `<dd>` elements in a `<div>` for styling - but only a `<div>`.
3. You can label them using ARIA.
4. They've been called "description lists", not "definition lists", since an HTML5 draft in 2008.

So this is valid:

```
<h2 id="credits">Credits</h2>
<dl aria-labelledby="credits">
  <div>
    <dt>Author</dt>
    <dd>Jeffrey Zeldman</dd>
    <dd>Ethan Marcotte</dd>
  </div>
</dl>
```

Here's a useful note from Adrian Roselli on screen reader support for description lists.

Tool: Mad House — Usborne Creepy Computer Games

Via Hacker News I learned that UK publisher Usborne published free PDFs of their 1980s Computer Books, some of which I remember working through on my Commodore 64 as a child.

These were so great! Beautifully illustrated books with fun projects made up of code you could type into your own machine.

I remember playing "Mad House" typed in from the 1983 book "Creepy Computer Games", so I fed that PDF into Claude and had it build an interactive version of that game in JavaScript and HTML:

```
Build a vanilla JS artifact that exactly recreates the game Mad House from this book, make sure it's mobile friendly and has a
suitable retro aesthetic

Credit the book title and link to https://usborne.com/us/books/computer-and-coding-books
```

Quote 2026-05-24

> The most frustrating failure mode right now is that people submit issues that are not in their own voice. They contain an observed problem somewhere, but it has been thrown into a clanker and the clanker reworded it and made a huge mess of it. Typically, it was prompted so badly that the conclusions produced are more often than not inaccurate but always full of confidence. The result is complete guesswork on root causes, fake-minimal repros, suggested implementation strategies, analogies to adjacent but often the wrong code, and long lists of error classes that might or might not matter. [...]

> So at least personally, I increasingly want issue reports to be condensed to what the human actually observed:
>
> 1. I ran this command.
> 2. I expected this to happen.
> 3. This happened instead.
> 4. Here is the exact error or log.

Armin Ronacher, on slop issues filed against Pi

Release: datasette-fixtures 0.1a0

One of the smaller features in Datasette 1.0a30 is this:

> New documented datasette.fixtures.populate_fixture_database(conn) helper for creating the fixture database tables used by Datasette's own tests, intended for plugin test suites.

This new plugin takes advantage of that API. You can try it out using `uvx` without even installing Datasette like this:

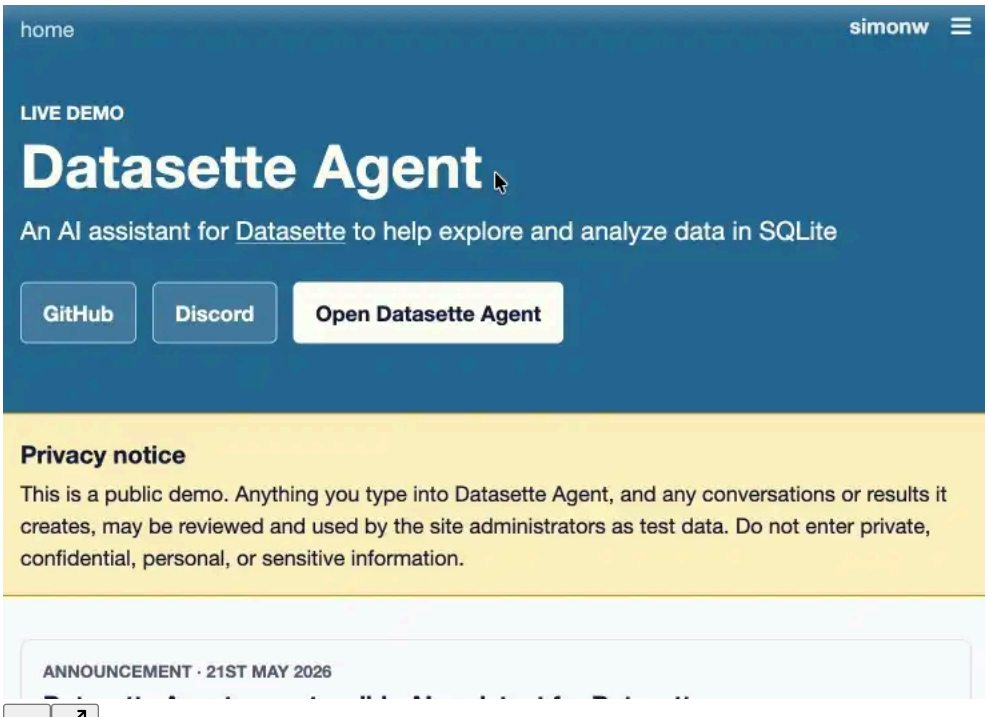
```
uvx --prerelease=allow \
  --with datasette-fixtures datasette \
  --get /fixtures/roadside_attractions.json
```

Which outputs:


```
{
  “ok”: true,
  “next”: null,
  “rows”: [
    {”pk”: 1, “name”: “The Mystery Spot”, “address”: “465 Mystery Spot Road, Santa Cruz, CA 95065”, “url”: “https://www.mysteryspot.com/”,
    {”pk”: 2, “name”: “Winchester Mystery House”, “address”: “525 South Winchester Boulevard, San Jose, CA 95128”, “url”: “https://winchest
    {”pk”: 3, “name”: “Burlingame Museum of PEZ Memorabilia”, “address”: “214 California Drive, Burlingame, CA 94010”, “url”: null, “latitu
    {”pk”: 4, “name”: “Bigfoot Discovery Museum”, “address”: “5497 Highway 9, Felton, CA 95018”, “url”: “https://www.bigfootdiscoveryprojec
  ],
  “truncated”: false
}
```

Release: datasette-agent 0.1a4

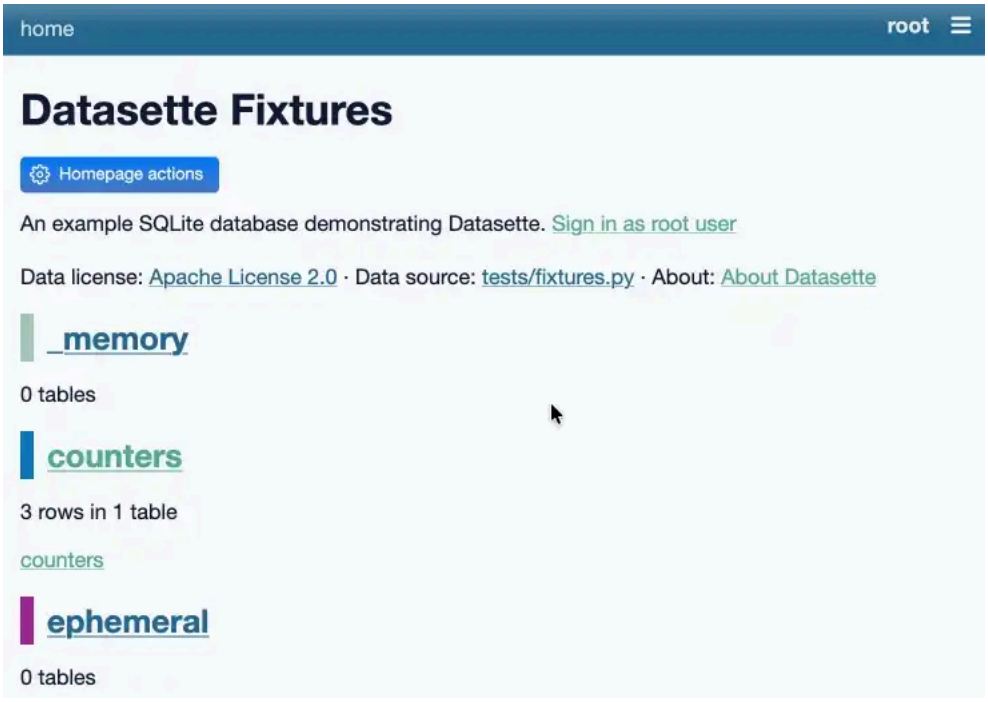
Taking advantage of the new makeJumpSections() JavaScript plugin hook added in Datasette 1.0a30, `datasette-agent` now presents this “Start a new agent chat” interface as part of the Jump to menu, any time you hit `/`:



You can try this out by signing into agent.datasette.io using your GitHub account.

Release: datasette 1.0a30

The big new feature in this alpha is a new customizable “Jump to...” menu, described in detail in The extensible “Jump to” menu in Datasette 1.0a30 on the Datasette blog. You can try it out by hitting `/` on latest.datasette.io - it looks like this:



The new jump_items_sql() plugin hook allows plugins to add their own items to the set that's searched by the plugin.

Quote 2026-05-26

> I cannot believe I'm saying this, but getting the literal Pope to canonize your product's specific technical limitations as a spiritual treatise is the single greatest act of vendor lobbying I have ever seen.

Corey Quinn, on Anthropic co-founder Christopher Olah's influence on *Magnifica Humanitas*

Quote 2026-05-26

> A lot of the emails I get from founders are now written in a hard-hitting journalistic style. I know they're written by AI, because no founder ever wrote this way before. And once you realize something is written by AI, it's hard not to ignore it.
>
> I have never knowingly finished reading an email signed by a human but written by AI. It feels like being lied to, and who would stand for that?
>
> [...] It makes me think less of the author. It means they can't write well unaided (or feel they can't), and that they're trying to trick me.
>
> It's not impressive to use AI to write stuff for you; any teenager can do that.

Paul Graham

Link 2026-05-26 Microsoft Copilot Cowork Exfiltrates Files:

The biggest challenge in designing agentic systems continues to be preventing them from enabling attackers to exfiltrate data.

In this case Microsoft Copilot Cowork (yes, that's a real product name) was allowing agents to send emails to the user's own inbox without approval... but those messages were then displayed in a way that could leak data to an attacker via rendered images:

> Because these messages can contain external images that trigger network requests to external websites, data can be exfiltrated when a user opens a compromised message sent by the agent.

Since OneDrive can create pre-authenticated download links, a successful prompt injection could cause those links to be leaked, allowing files to be downloaded by the attacker.

Link 2026-05-26 The pressure:

Daniel Stenberg on the unprecedented level of pressure the `curl` team are facing right now thanks to the deluge of (credible) AI-assisted security issues being reported.

> The rate of incoming security reports is 4-5 times higher than it was in 2024 and double the speed of 2025 -- meaning that **on average we now get more than one report per day**. The quality is way higher than ever before. The reports are typically *very* detailed and long. [...]
>
> For the first time in my life, my wife voiced concerns about my work hours and my imbalanced work/life situation. I work more than I've done before, but the flood keeps coming. [...]

> This is a never-before seen or experienced pressure on the curl project and its security team members. An avalanche of high priority work that trumps all other things in the project that is primarily mental because we certainly *could* ignore them all if we wanted, but we feel a responsibility, we have a conscience and we are proud about our work.

The good news is that `curl` is a very solid piece of software, so the vulnerabilities people are finding tend not to be of high severity:

> What is also a good trend: almost no one finds *terrible* vulnerabilities. All vulnerabilities found the last few years in curl have *all* been deemed severity LOW or MEDIUM. I'm not saying there won't be any more HIGH ever, but at least they are rare. The most recent severity high curl CVE was published in October 2023.

Quote 2026-05-27

> PICARD: Data, shields up
>
> DATA: Brilliant! Shields can reduce damage we sustain. Not immunity. Not hubris. Just prudence. It's not precaution—it's strategy.
>
> [camera shakes]
>
> WORF: HULL BREACHES ON NINE DECKS
>
> DATA: Here's what happened: you told me to raise shields, and I didn't

Kyle Ferrana, @KyleTrainEmoji

Link 2026-05-27 sqlite AGENTS.md:

SQLite gained an AGENTS.md file five days ago - but it's not intended for their own development, it's presumably aimed at people who are pointing agents at the SQLite codebase. It includes:

> SQLite does not accept pull requests without prior agreement and/or accompanying legal paperwork that places the pull request in the public domain. However, the human SQLite developers will review a concise and well-written pull request as a proof-of-concept prior to reimplementing the changes themselves.
>
> SQLite does not accept agentic code. However the project will accept agentic bug reports that include a reproducible test case. Patches or pull requests demonstrating a possible fix, for documentation purposes, are welcomed.

The most recent commit to that file removed "(currently)" from "SQLite does not (currently) accept agentic code", with the commit message "Strengthen the statement about not accepting agentic code".

Meanwhile the SQLite forum was being flooded with so many AI-generated bug reports - of varying quality - that they've now split those off into a new SQLite Bug Forum. D. Richard Hipp is resolving issues on there with a flurry of commits to the codebase.

Release: llm-anthropic 0.25.1

- New model: Claude Opus 4.8 (`claude-opus-4.8`).
- New `-o fast 1` option for fast mode, for organizations with that feature enabled on their account.
- Default max_tokens for each model now defaults to that model's maximum output rather than 8,192. #72

See also my notes on Opus 4.8 - I used this new release of `llm-anthropic` to generate the pelicans.

Note 2026-05-29

The most interesting thing about Anthropic's $65B Series H announcement is this line (emphasis mine):

> Since our Series G in February, adoption has continued to grow across global enterprise customers, and our run-rate revenue crossed **$47 billion** earlier this month.

Anthropic have made a bit of a habit of sharing their "run-rate revenue" in this kind of announcement, which is an annualized projection of their current revenue - typically calculated by taking the most recent month and multiplying by 12.

Earlier this year:

- Apr 6, 2026 in Anthropic expands partnership with Google and Broadcom: "Our run-rate revenue has now surpassed **$30 billion**—up from approximately **$9 billion** at the end of 2025."
- Feb 12, 2026 in Anthropic raises $30 billion in Series G: "Today, our run-rate revenue is **$14 billion**, with this figure growing over 10x annually in each of those past three years."

I had Claude Opus 4.8 make me this chart using Matplotlib (Claude: "a data line chart is more straightforward matplotlib work—not really a design piece"):

Back in April Axios CEO Jim VandeHei wrote that he could not find “any company — in any industry, in any era — that has scaled organic revenue this quickly at this level as Anthropic” - and that was when they were at a paltry $30 billion.

(Also in Axios today is an anonymously sourced note that “An AI consultant tells Axios one of their clients recently spent half a billion dollars in a single month after failing to put usage limits on Claude licenses for employees” - times that by 12 and you get an extra $6 billion in annualized run-rate!)

Ed Zitron was extremely skeptical of that $30 billion number - I wonder if his skepticism will update for the new $47 billion figure.

I’ve seen a few people dismiss this as untrustworthy, because the numbers come from Anthropic. That doesn’t hold up: these numbers were included in announcements of their fundraises, and lying to investors who just put in $65 billion would be securities fraud. They’re even less likely to lie given that the real numbers will no doubt come out in their S-1 when they file for their IPO.

If you find this newsletter useful, please consider sponsoring me via GitHub. $10/month and higher sponsors get a monthly newsletter with my summary of the most important trends of the past 30 days - here are previews from January and February and March.

Tailscale Kubernetes Operator v1.98.4

A new release of the Tailscale Kubernetes Operator is available. For guidance on installing and updating, refer to our installation instructions.

- The operator no longer fails to perform token exchanges when using workload identity
- Ingress & Egress ProxyGroup pods now correctly clamp their MTU values

Google is in a strange spot right now. They've got arguably the deepest research bench in the industry, their own custom silicon, and effectively unlimited money - and yet most developers I talk to barely touch Gemini day-to-day. The recent Google I/O announcements crystallised a lot of what I find confusing about their AI strategy, so I wanted to write down where I think they actually stand.

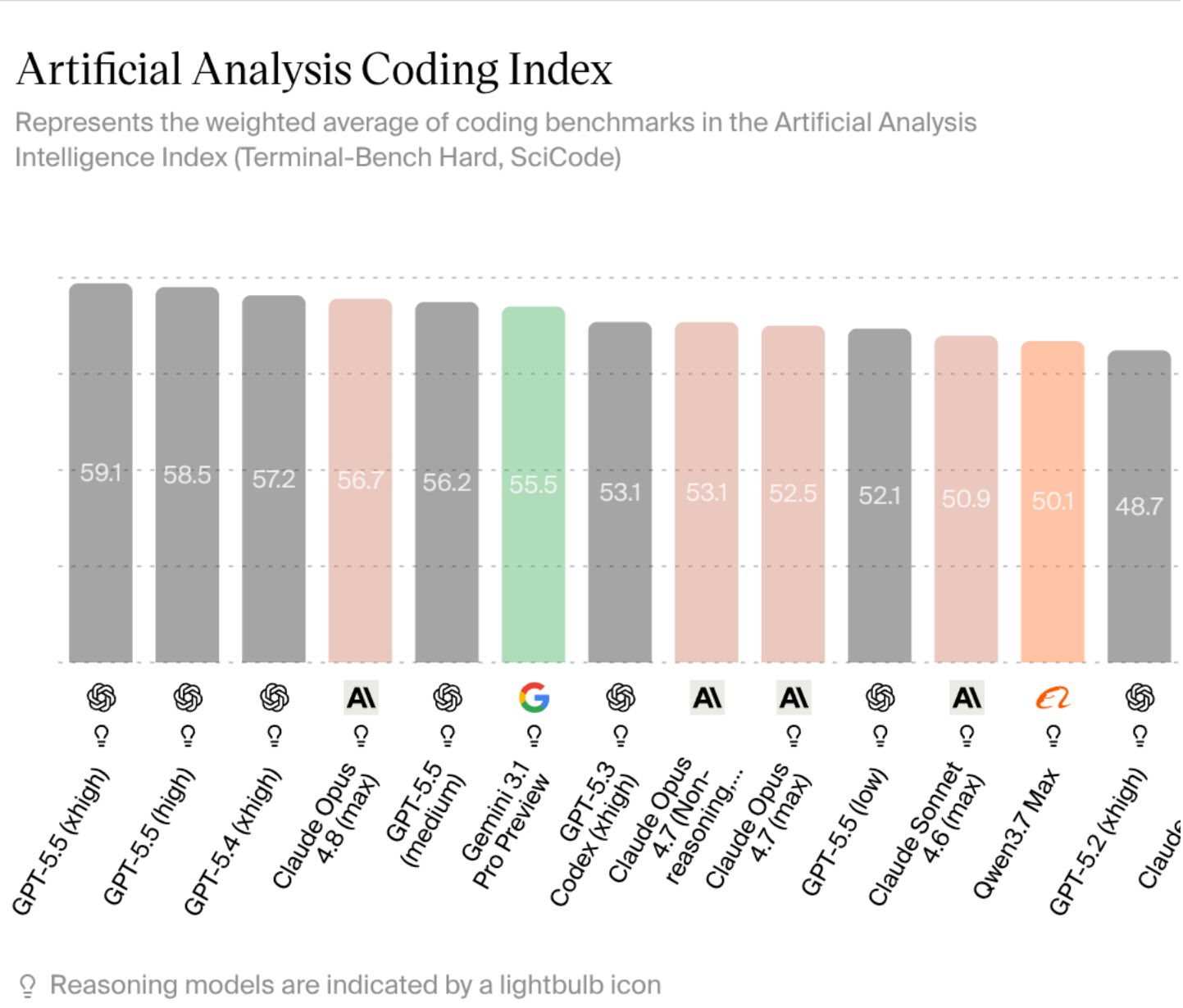
The state of play

The consensus seems to be that currently Anthropic and OpenAI are very much in the lead for frontier model intelligence, with each of those two labs trading blows every month. This may change in the near future - if Anthropic releases Mythos-class models that OpenAI doesn't have an answer to - but right now I think most practitioners would agree that GPT5.5 and Opus 4.8 are roughly in the same ballpark.

After that, you have Google, with Gemini 3.1 Pro being in benchmarks ahead of the Chinese models but behind the flagship Anthropic/OpenAI models. In my personal experience though I've had better results from the best-in-class Chinese models (GLM 5.1 and Qwen 3.7) than Gemini 3.1 Pro at software engineering tasks.

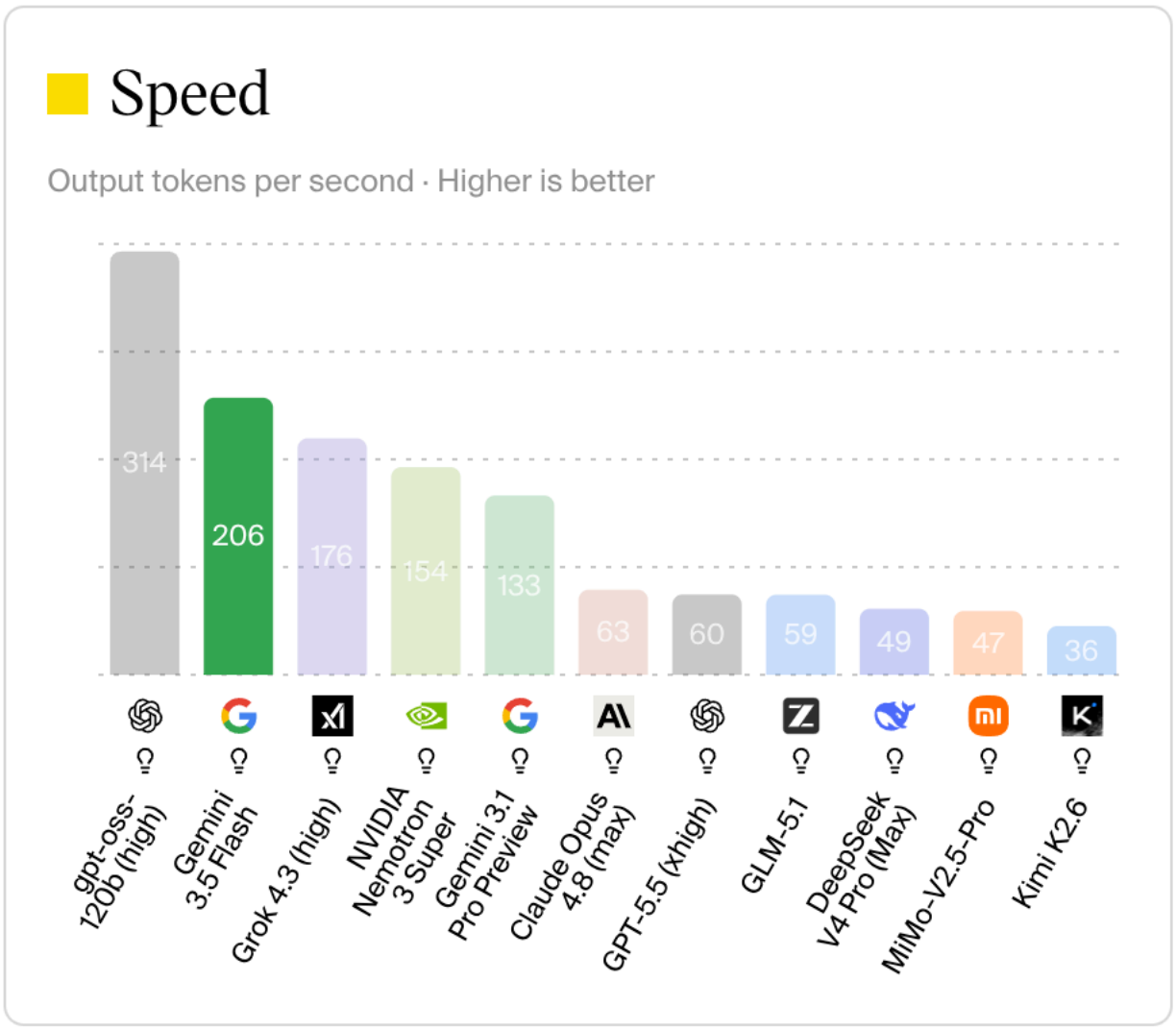
Gemini 3.5 Flash is confusing

The main model announcement at Google I/O was Gemini 3.5 Flash. The benchmarks of it were underwhelming at coding:



Gemini 3.5 Flash on the Artificial Analysis Coding Index - solidly mid-pack. Source: Artificial Analysis.

However, the model is *super fast* - roughly 4x faster in tokens per second than the aforementioned Anthropic/OpenAI models:



Output tokens per second - Gemini 3.5 Flash at 206 t/s, far ahead of Opus 4.8 and GPT-5.5. Source: Artificial Analysis.

This definitely is really interesting development, especially for user facing applications which can appear very sluggish to users.

But - the big but - is the *huge* price increase they announced - 3x more expensive than the previous flash release. At $9/MTok it is vastly more expensive than the best in class Chinese models, and I'm struggling to see where this fits - if you want best in class intelligence you pay the extra for Opus/GPT5.5, if you want cheap but not-as-clever the Chinese models fit the bill well. The risks around Chinese models are somewhat overplayed in my opinion - you can self-host a lot of them, or use US-based inference providers via OpenRouter.

Is this model really for Google itself?

Having said all that, perhaps really this model isn't designed for external use in the same way that the OpenAI/Anthropic models are. Clearly Google consumes an enormous amount of tokens internally - for all their products like AI mode, Gmail, etc.

If you look at it that way, the model makes far more sense. The speed of the model really matters for a lot of the Google use cases - AI mode is very user driven and Google knows better than anyone that speed really matters. And the actual serving cost Google pays is almost certainly a fraction of the external facing price, so that becomes irrelevant.

The most interesting part of this story though, is this excellent comment on Hacker News from someone that estimated the size of the model and the fact that it should run on *one* TPU 8i card (Google's latest custom inference hardware).

This does give Google a huge advantage. They are the only frontier lab that (currently) designs its own AI hardware. While other labs certainly optimise their models *to* the hardware, and also no doubt have a lot of say in *driving* the Nvidia/AMD roadmaps to their specifications, the model teams and hardware teams in Google almost certainly collaborate to a far greater level than the other labs.

This *really* matters. If you have a very good steer on upcoming hardware you know the right size of models to target training runs to aim for. And equally, research from Google Deepmind can go straight into the hardware roadmap without any negotiations. [1]

It'll be very interesting to see how this continues to develop. Inference efficiency will be the key driver to actual unit economics in AI, and Google may develop an outsized lead in this.

Coding agents

The one real weakness I think Google has though, is their confusing and incoherent strategy on coding agents. While Anthropic has Claude Code, and OpenAI has Codex, in true Google style they have ended up with a smorgasbord of tools.

There is currently Antigravity, Jules, Gemini Code Assist, Gemini CLI and AI Studio all doing slightly different things. This doesn't include some *other* agentic SWE tools they have for specialised purposes (like Android Studio).

They announced that Gemini CLI is being discontinued and folded into Antigravity, but I very rarely come across *any* developer using Google-based SWE tooling.

This is a huge issue for Google - there is no doubt that Claude Code and Codex is producing a lot of very detailed telemetry and training data that can be used to improve further models. Without this being resolved, Google does have an extreme weakness in the fastest growing - at least revenue-wise - segment of AI.

While I definitely wouldn't write Google off - they do have enormous structural advantages in other areas - I get the feeling that because Google has such a bespoke internal software development workflows[2] their isolation from what "the rest of the industry" does in software is so large it's perhaps hard for them to really reason about agentic tooling for the rest of the industry.

So, what's going on with Gemini?

My read is that Google is playing a genuinely different game to OpenAI and Anthropic. Gemini 3.5 Flash only looks strange if you assume it's meant to win the same race - priced and tuned for Google's own gigantic internal token consumption, with the TPU advantage baked in, it makes complete sense. Where they're actually behind is the developer-facing surface: a confused tangle of coding tools and an org that struggles to reason about how the rest of us build software. If Google sorts out the agent story, the structural advantages underneath - the silicon, the research, the integration - could make them very hard to beat. That's a big if. But I wouldn't bet against them.

---

1. While it's hard to say if there was any truth in this - or it was just a negotiation strategy - there were rumours of OpenAI being unhappy with direction/progress Nvidia was making earlier this year: https://finance.yahoo.com/news/sam-altman-pushes-back-report-213000823.html ↩

2. Google engineers have an enormous amount of home built/custom/internal tooling that is uncommon outside of Google-scale companies. They use different source control, build tooling, testing infrastructure and build deployment to the rest of the industry - for very good reasons! But this stack is absolutely overkill for 99% of companies, and when you are used to thinking about SWE at Google scale I suspect it is very difficult to reason how people build software *outside* of that ecosystem. ↩

Coding agents today have a massive spending problem. Every request, whether you're designing system architecture or writing a single-line docstring, often gets routed to the same expensive frontier model. The result: unnecessary token usage, higher inference costs, and little awareness of task complexity or budget constraints.

This high cost stems from a "one-size-fits-all" approach to model usage, where premium frontier models are utilized for trivial tasks that don't require such intensive reasoning effort. In multi-agent workflows, where orchestrators delegate work to specialized subagents, this lack of discrimination frequently leads to runaway costs and opaque failure modes. Without intelligent routing, developers can essentially be forced into closed-provider lock-in and high API usage fees, which quickly escalate during exploratory building phases.

DigitalOcean Inference Router, now in Public Preview, was built to solve this problem by dynamically routing requests to the right model for the job. As part of DigitalOcean's AI-Native Cloud, it gives developers a unified way to control, optimize, and evaluate AI inference across models. And as of today, you can access it through OpenCode, the open-source AI coding agent, in as little as a few seconds.

What is an Inference Router?

An Inference Router is the auto-mode pattern engineers are used to, but with deliberate control over the tradeoffs that matter: latency, cost, and output quality. Rather than statically pointing your coding agent to a single model, an Inference Router can analyze each request and route it to the model best suited for that specific task. Not the *most* powerful model available, but the *right* model. That distinction is what drives real savings without compromising on your desired quality of output.

To use DigitalOcean's Inference Router: Create an Inference Router from the router catalog—pick a preset or build a custom router via the API or UI. No GPU management, no infrastructure to run. Use it by setting "model": "router:your-router-name" in any OpenAI-compatible API call.

What Changed for OpenCode

OpenCode has become one of the most popular AI coding harnesses on GitHub, earning over 160,000 stars by embracing a simple idea: developers should not be locked into a single model provider. Its rise has shown a demand for provider agnostic AI use cases. At Deploy 2026, Tyler Gillam - a core engineer on Inference Router - demoed our integration live on stage, showing exactly how OpenCode and Inference Router work together to make intelligent model selection decisions in real time. If you want to see it before diving in, the full recording is linked at the bottom of this post.

Previously, integrating DigitalOcean models into OpenCode meant manually editing your `opencode.json`, adding each model by hand, a list that would be outdated within weeks given the pace of new model launches. So, we built a native OpenCode integration that supports Inference Routers and DigitalOcean Serverless Inference models right out of the box.

Now you can run the following steps:

1. Launch OpenCode (desktop, web or TUI) and run `/connect`
2. Select **Login with DigitalOcean**
3. Your Inference Routers are shown in the Model Selection tab

That's it. You're plugging directly into a routing layer that's already helping to make the cost & quality tradeoff decisions for you based on your stated needs — with our purpose-build Software Engineering preset.

Beyond Coding Agents

This integration is part of a broader effort to bring DigitalOcean’s Inference Engine into the tools developers already use, while continuing to invest in open source and upstream contributions. OpenCode is one example of that direction.

The goal is to make intelligent, cost-aware model routing the default for coding agents, not something you have to manually configure and hope for the best. As the OSS model landscape keeps improving, routing intelligence will become more valuable, not less. The gap between “frontier” and “good enough” is closing fast, and developers who take advantage of routing will consistently come ahead on both desired quality and cost.

If you’re using OpenCode, try `/connect` today. If you want to dig deeper on what Inference Router is and how it works, the full documentation is available below.

Inference Router Resources:

- How We Built DigitalOcean Inference Router
- Inference Router Documentation
- OpenCode DigitalOcean Install
- InferenceRouter OpenCode Deploy 2026 Demo