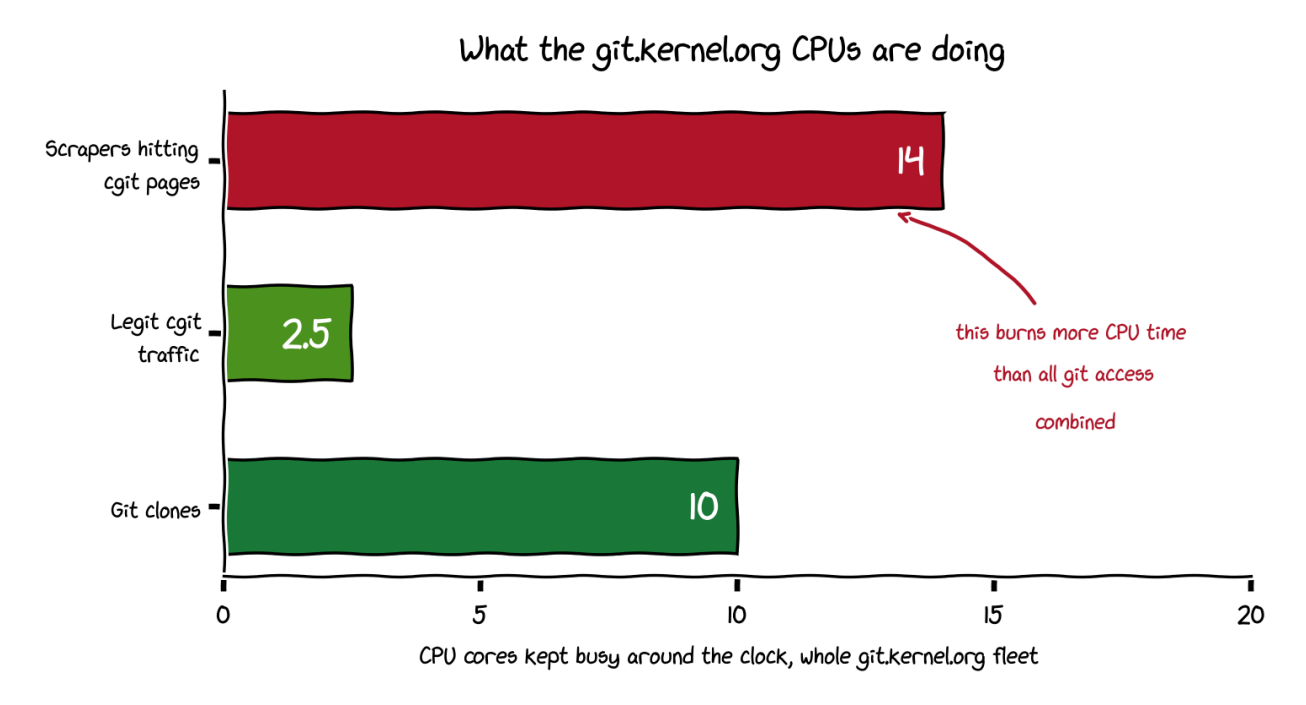
Creepy crawlies

August 29, 2026

You've probably heard me complain about the “AI crawlers” before, but now I actually have some hard numbers I can put up to show their impact. In a few words, it's bad enough to create a constant “background radiation” of system load, permanently tying up a chunk of capacity spent on producing output that is only useful for a single purpose — feeding a learning model.

TL;DR: we spend more CPU cycles rendering commits for scrapers than we spend on all other kinds of legitimate access, including git clones. At any one time, across 5 geo-distributed nodes, there are 14 CPU cores doing nothing but rendering git commits as html.



Why is git.kernel.org “interesting” to crawlers

Linux development happens in the open — from git repositories you can clone, to discussion archives you can follow in real time. To a large language model, this is a goldmine of learning data, because all of this is not only immediately available, but is easy to filter in order to guarantee pure unadulterated pre-AI content. Training an LLM on content produced by the LLM gives it the equivalent of a digital prion disease, so when a source is guaranteed to be LLM-free, like the entire history of kernel commits, it's worth its weight in gold as a source of training data.

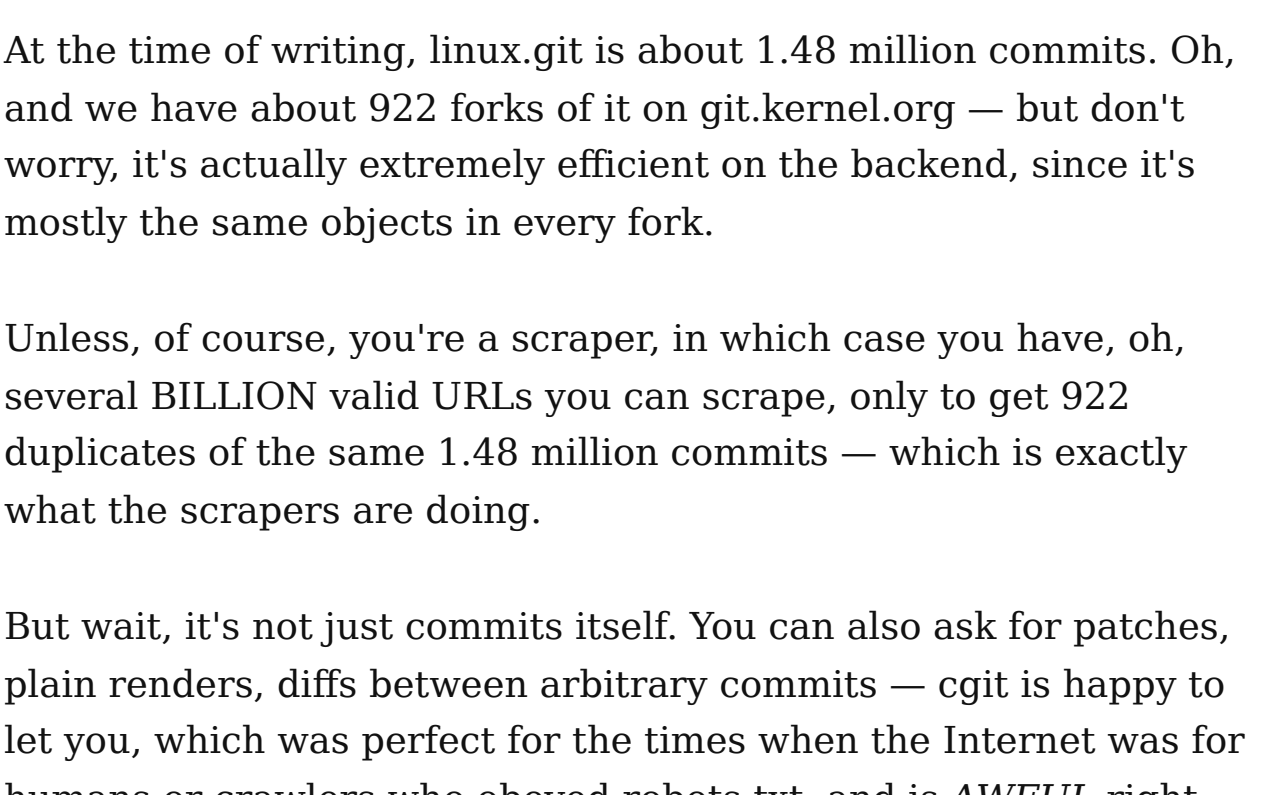
The stupidest way of doing it

We make almost everything clonable, because hey — we may not be around forever, so here — clone the repos. Also, clone the archives. Grab a copy just so we're not the only ones who have it all. Seriously, it's just a “git clone” away — and then you'll have the whole history.

For example, did you know you can clone the entirety of LKML and then do whatever you want with it? It's just git repos all the way down.

So, you'd think that something that pretends to be “Artificial Intelligence” would use the most efficient way of using our data for training purposes, right? Clone the repos, walk every commit. Done.

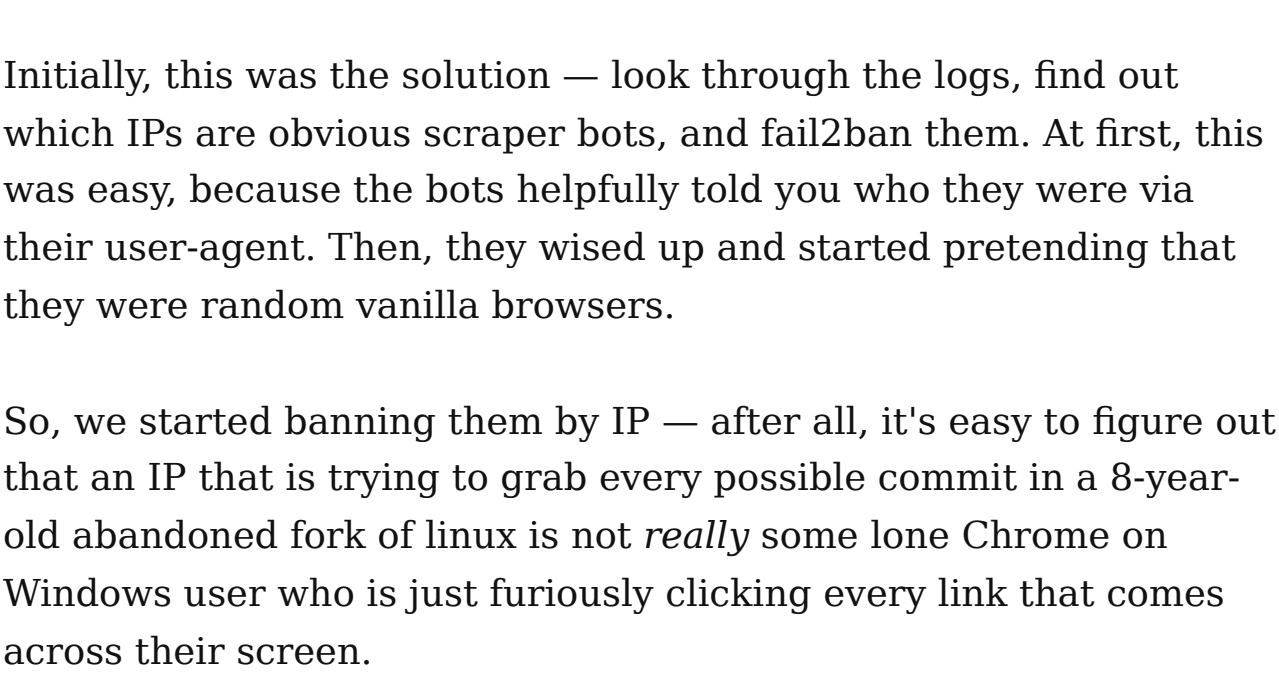
But no, let's in fact choose the stupidest possible way of doing it — by rendering everything as HTML commit by commit and then parsing it.



At the time of writing, linux.git is about 1.48 million commits. Oh, and we have about 922 forks of it on git.kernel.org — but don't worry, it's actually extremely efficient on the backend, since it's mostly the same objects in every fork.

Unless, of course, you're a scraper, in which case you have, oh, several BILLION valid URLs you can scrape, only to get 922 duplicates of the same 1.48 million commits — which is exactly what the scrapers are doing.

But wait, it's not just commits itself. You can also ask for patches, plain renders, diffs between arbitrary commits — cgit is happy to let you, which was perfect for the times when the Internet was for humans or crawlers who obeyed robots.txt, and is *AWFUL* right about now, because we can generate 1.2 *METRIC BAJILLION* valid URLs just for a single fork of linux.git.



Block them

Initially, this was the solution — look through the logs, find out which IPs are obvious scraper bots, and fail2ban them. At first, this was easy, because the bots helpfully told you who they were via their user-agent. Then, they wised up and started pretending that they were random vanilla browsers.

So, we started banning them by IP — after all, it's easy to figure out that an IP that is trying to grab every possible commit in a 8-year-old abandoned fork of linux is not *really* some lone Chrome on Windows user who is just furiously clicking every link that comes across their screen.

The bots then started fanning out to entire subnets, but this was still meh, because obviously an IP coming from Google Compute is just *pretending* to be a Firefox user. Banning the whole ASN was justified, even if this occasionally caught a random legitimate instance trying to automate link checking in commits.

Enter... your TV?

And... that's when things turned really, *really* ugly. Suddenly, the crawlers were coming from millions of random residential or mobile IPs, all pretending to be random modern browsers. An IP like that would make 4-5 requests and then never show up in the logs again. There was no point in banning them, because by the time you figured out that they were bots, they were already done with you. You just needlessly ballooned your firewall ruleset by adding IPs that would never be back.

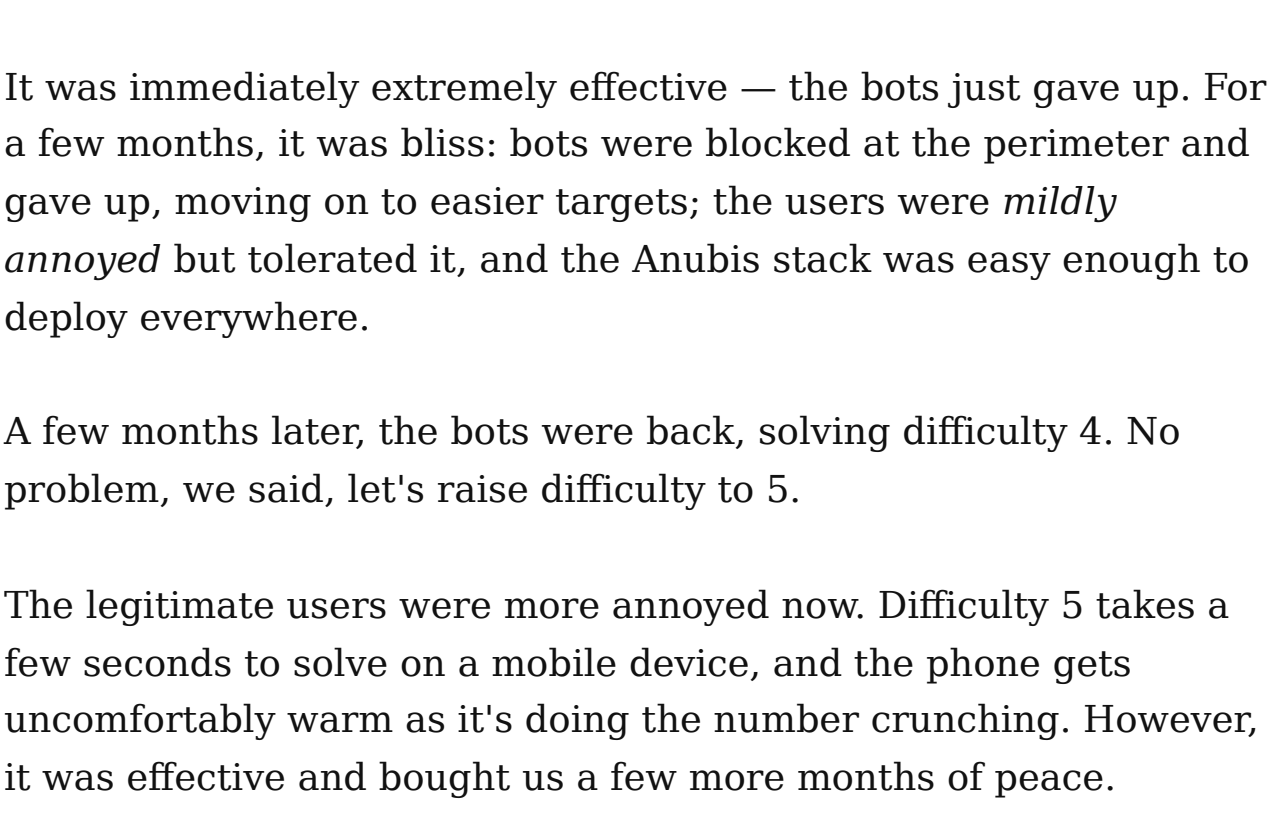
They descended like swarms of locust, hit hard and fast until the system fell over and then moved on to the next target until you recovered. Then, they returned. Rinse. Repeat.

They still do that — welcome to the wonderful world of “proxy SDK monetization.” It's big business, and your TV is probably doing it.

Make them pay

When this first became a problem, oh, about a year ago, we naively thought that there was a way to make it stop. Just make the bots perform a task that would flip the economy of the whole thing upside-down by making them burn some cycles doing throwaway math. Like, calculate what string, when combined with their own IP and a secret we provide, would generate a sha256 sum with 4 leading zeroes.

In other words, we put Anubis in front of everything.



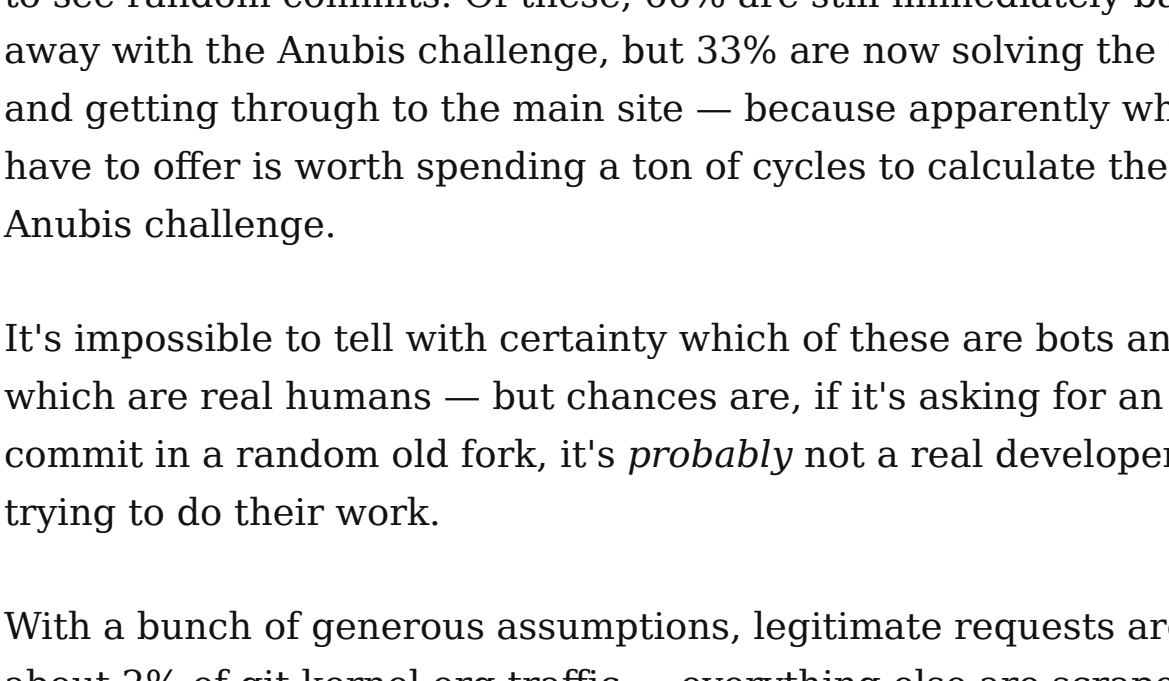
It was immediately extremely effective — the bots just gave up. For a few months, it was bliss: bots were blocked at the perimeter and gave up, moving on to easier targets; the users were *mildly annoyed* but tolerated it, and the Anubis stack was easy enough to deploy everywhere.

A few months later, the bots were back, solving difficulty 4. No problem, we said, let's raise difficulty to 5.

The legitimate users were more annoyed now. Difficulty 5 takes a few seconds to solve on a mobile device, and the phone gets uncomfortably warm as it's doing the number crunching. However, it was effective and bought us a few more months of peace.

Then... the bots started solving difficulty 5.

Where we are now

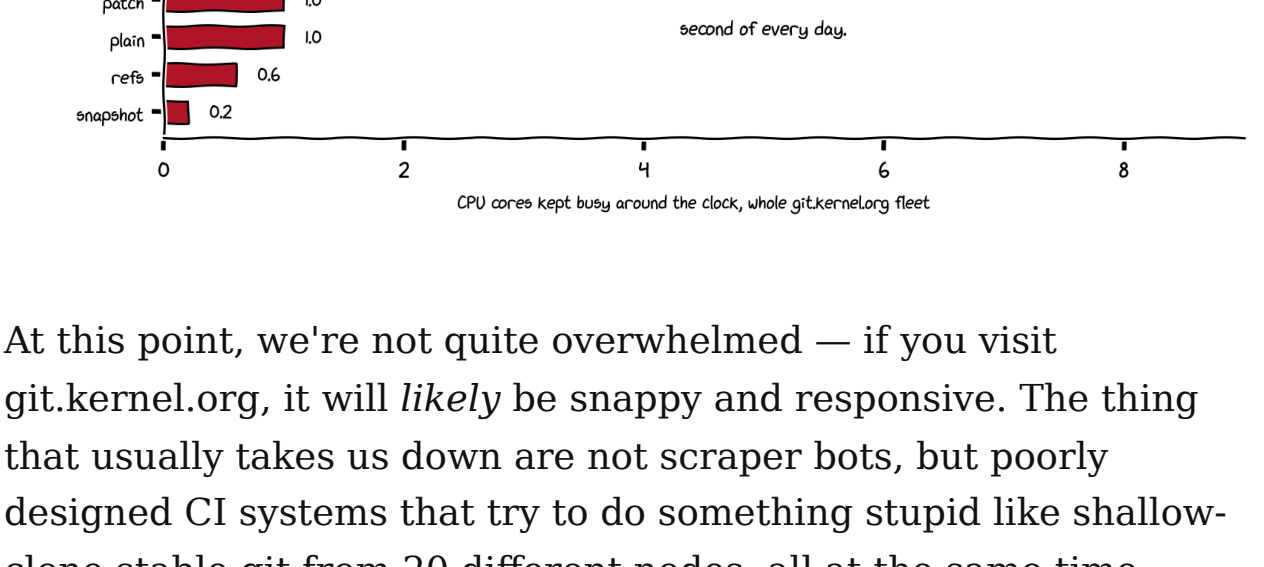


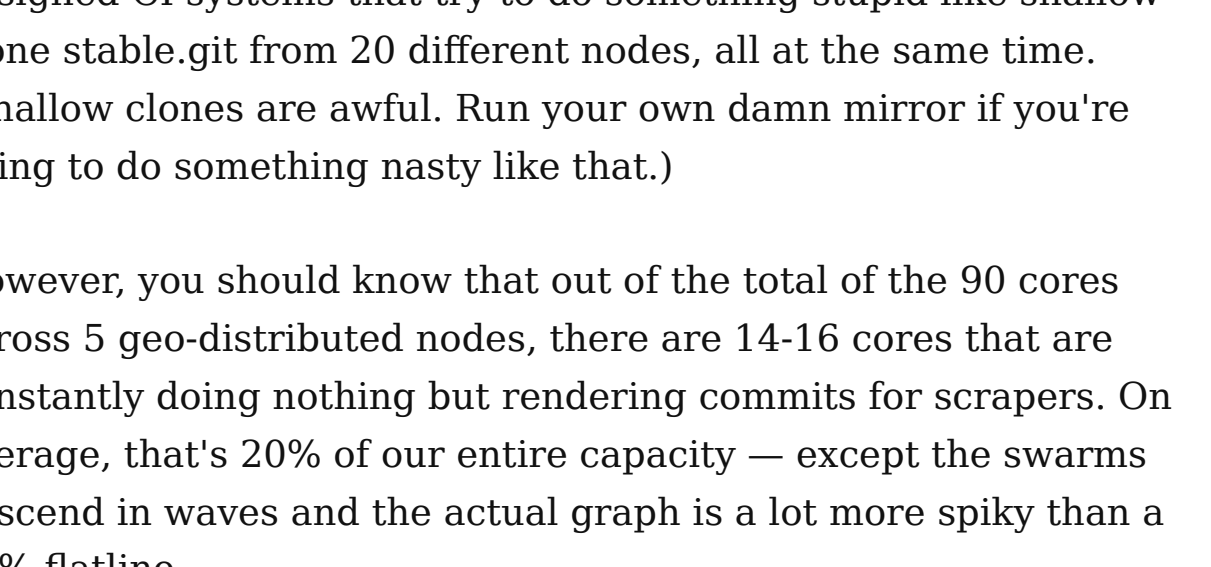
Today, git.kernel.org receives about 6M daily requests demanding to see random commits. Of these, 66% are still immediately batted away with the Anubis challenge, but 33% are now solving the math and getting through to the main site — because apparently what we have to offer is worth spending a ton of cycles to calculate the Anubis challenge.

It's impossible to tell with certainty which of these are bots and which are real humans — but chances are, if it's asking for an old commit in a random old fork, it's *probably* not a real developer trying to do their work.

With a bunch of generous assumptions, legitimate requests are only about 2% of git.kernel.org traffic — everything else are scrapers.

How bad is it?





At this point, we're not quite overwhelmed — if you visit git.kernel.org, it will *likely* be snappy and responsive. The thing that usually takes us down are not scraper bots, but poorly designed CI systems that try to do something stupid like shallow-clone stable.git from 20 different nodes, all at the same time. (Shallow clones are awful. Run your own damn mirror if you're going to do something nasty like that.)

However, you should know that out of the total of the 90 cores across 5 geo-distributed nodes, there are 14-16 cores that are constantly doing nothing but rendering commits for scrapers. On average, that's 20% of our entire capacity — except the swarms descend in waves and the actual graph is a lot more spiky than a 20% flatline.

Where does that leave us?

Unclear. Maybe the AI bubble bursts and we suddenly have a lot fewer entities out there trying to train their models. Alternatively, maybe they smarten up and stop consuming our data in the dumbest way possible.

In terms of what we're doing, we're turning off features to reduce the number of crawlable URLs and to gate off actions that are expensive for us to run. Expect to lose some functionality, at least when accessing our resources anonymously. Trust me, we hate it just as much as you, but at this point it's a necessity.

Worst of all, there are no simple solutions to the problem. Companies offering custom “AI” models still pop up daily, all of them hungry for training data. App makers are still looking for ways to turn a profit, so they will continue to turn your household appliances into attack vectors.

That said, we promise to still offer all of our data for download to anyone who asks. You just may have to jump through more hoops to get it.

Sorry. (Oblig. Canadian thing to say.)