Episode Book

EPISODE #6334

Why Fast Backups Trigger Rate Limits

PUBLISHED	28 October 2026 at 17:25 UTC
DURATION	28:58
LISTEN	myweirdprompts.com/episode/api-rate-limits-backup-quotas-13832/
HOSTS	Corn, Herman, Hilbert
SCRIPT MODEL	openai/gpt-6.1-sol
VOICES	chatterbox-regular (chatterbox-local)
PIPELINE	V5.25
SOURCES	19 cited by the research (listed at the end)
BOOK GENERATED	29 October 2026 at 08:41 UTC

SYNOPSIS

API rate limits can stall automated backups even when accessing your own files. Looking at tools like rclone and the gws command line interface, this discussion explores how tiered quotas, token buckets, and hidden backend protections shape API behavior. Learn why method weights penalize recursive file listing, how shared project quotas mask individual activity, and why simply retrying with exponential backoff fails when an agent confuses rate limits with hard operational caps.

TRANSCRIPT



Corn

Your backup stops halfway through, the terminal says too many requests, and suddenly saving your own files looks suspicious. Daniel ran into this backing up Google Workspace with a command line tool. His question is what happens as AI agents put that kind of automation within everybody's reach. How does the service decide you're asking too fast, what does a decent client do when it gets refused, and can legitimate agents identify themselves without looking like spammers?



Herman

The first distinction matters immediately. An HTTP four twenty-nine means the service is pushing back on traffic. It does not necessarily mean anyone has decided you're malicious. You might be competing with other perfectly legitimate work, or hitting a limit your dashboard doesn't display.



Corn

Innocent, but still not backed up.



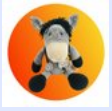
Herman

Right. The user sees a failed job. The defender sees a stream of operations against shared infrastructure. Neither gets to inspect the other's intentions directly.



Corn

Also, Herman, I predict you're about to admire the elegance of the limiter before we've recovered a single document.



Herman

I admire an elegant limiter that tells you what it limited. A mysterious refusal with a cheerful retry loop gets much less admiration.



Corn

Your standards are rising.



Herman

Living with your software reviews has consequences.



Herman

Rate limiting starts with a counter and a bucket to count against. That bucket might belong to an account, an IP address, an application project, or some combination. Each operation consumes capacity. Once the allowed capacity is exhausted, the server delays or rejects more work until capacity returns.



Corn

The uncomfortable bit is that ordinary use has changed shape. Reading documents through a browser involves pauses. A backup doesn't stop to consider a paragraph. It lists, checks, downloads, repeats. Give that job to an agent and the person issuing it still feels like they're making one request.



Herman

Whereas the API may see thousands. Older descriptions of attack traffic sometimes contrast a compromised machine issuing more than twenty web requests a second with much lower legitimate rates. That's an illustration, not a universal threshold. A legitimate command line client can exceed that pace quite comfortably.



Corn

Twenty requests a second is now somebody being tidy.



Herman

The speed doesn't establish intent. More importantly, honest traffic can still exhaust a service. If everybody backed up a large account at once, the infrastructure would have a capacity problem even if every request came with a handwritten apology.



Corn

I accept the need to ration. I object when the ration depends on a secret unit and the refusal says something unrelated. A backup is especially irritating because the files already belong to you. You're not trying to collect somebody else's website.



Herman

Ownership establishes why you should be allowed access. It doesn't make enumeration, permission checks, database queries, or outgoing bandwidth free. That's why authentication and capacity controls coexist.



Corn

Fair. Though the interface encourages the confusion. You sign in, approve access, and the tool says it's ready. Nobody says that ready means ready to discover a second permission structure made of counters.



Herman

Would you prefer a slower backup that finishes overnight, Corn, or a faster one that occasionally pauses unpredictably?



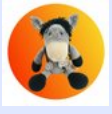
Corn

Overnight, if it gives an honest completion estimate and resumes properly. Unpredictable pauses are tolerable. Unpredictable restarts aren't. I don't want an agent spending the night enthusiastically rediscovering the first folder.



Herman

That distinction is going to matter when we get to retries. First, the counters themselves can explain why a tool that barely seems busy is already in trouble.



Herman

A fixed window limiter counts work within a defined interval. A sliding window tracks a moving interval, avoiding some boundary effects. A token bucket allows a limited burst, then replenishes capacity at a steady rate. These designs behave differently even when



Herman (continued)

their advertised average allowance sounds identical.



Corn

Meaning a minute's allowance doesn't necessarily mean I can spend it in the first second.



Herman

Correct. A service can permit a sustained rate while rejecting an abrupt burst. It can also maintain several counters simultaneously. Your account has one allowance, the application's project has another, and the backend may enforce additional protections.



Corn

That creates a nasty household problem. My job slows down because your job is busy, but my error message makes it sound personal.



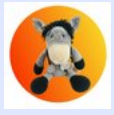
Herman

Shared project quotas really can do that. The request is authenticated as yours, yet it consumes a pool other users of the same application also draw from. A shared network can create a different version through IP limits. Separate identities don't guarantee separate capacity.



Corn

The technical term is apparently togetherness.



Herman

Drive makes the layered accounting unusually concrete. For projects under the quota scheme introduced in May, the published project allowance is one million quota units per minute. The per-user allowance within that project is three hundred twenty-five thousand per minute.



Corn

Those sound enormous until you tell me what a unit buys.



Herman

Listing files costs one hundred units per API call. A metadata get costs five. The download method costs two hundred. Those are method weights, not a count of files or a direct measure of bytes.



Corn

Looking around costs twenty times as much as looking at something.



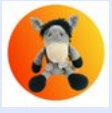
Herman

Per call, yes. A listing call can return a page containing multiple files, so you mustn't translate that into one hundred units per file. Page size, folder structure, requested fields, and the tool's traversal strategy all affect the work.



Corn

That correction changes the mental picture. A well-designed listing can cover plenty of files, while a tool that repeatedly walks folders spends capacity proving it has folders. The backup hasn't failed because the documents are enormous. It may have failed while assembling the shopping list.



Herman

A method that updates a file costs fifty units, too. Read versus write is not a sufficient guide to cost. Services assign weights to reflect their own resource model, and that model won't necessarily match what feels expensive to you.



Corn

I would have guessed downloading dominates everything.



Herman

Bytes matter separately. The published scheme includes one terabyte of egress per day per project. There's also a daily billing threshold of four hundred million quota units that the documentation says cannot be raised. A billing threshold and a short-term rate allowance are different concepts.



Corn

More counters. Fewer documents.



Herman

Older projects can retain older quotas, which is why comparing two people's screenshots without checking their project history can be misleading. You can both be describing your setup accurately and still have different allowances.



Corn

This is where an agent can make confidence actively unhelpful. It sees a large published quota, observes much less activity, and concludes the service must be broken. Then it increases concurrency to make progress.



Herman

Or treats the error as an instruction to repeat the operation without changing anything. The published quota is only part of the picture. Drive explicitly says additional backend rate checks can generate a four twenty-nine response.



Corn

The dashboard can be green while the backup is stationary.



Herman

Yes. A dashboard for a project allowance doesn't promise visibility into every backend constraint. Local hotspots, shared resources, or protective checks can still matter. I don't know which internal check fired in any particular user's case, and a quota chart alone can't



Herman (continued)

tell us.



Corn

That uncertainty should be in the error response, rather than something users reconstruct from three evenings of reducing settings.



Herman

Let me ask you something. If the service disclosed every protective threshold, would you consider that an unconditional improvement?



Corn

Not every abuse threshold. An attacker shouldn't receive a detailed map of how to stay invisible. But an authenticated customer should be told whether the next useful action is waiting, reducing concurrency, changing permissions, or asking for a different allowance. That's operational information, not necessarily the burglar alarm's wiring diagram.



Herman

There's a genuine balance there. Detailed detection logic can help attackers adapt, while vague feedback makes legitimate clients waste capacity. A service can disclose the category and expected recovery behavior without publishing its entire abuse model.



Corn

Especially when the HTTP number is already doing too many jobs.



Herman

Drive quota errors can arrive as four oh three with a user rate limit message. Additional backend throttling can arrive as four twenty-nine. But four oh three also covers unrelated permission problems. The client needs the structured reason, not just the status number.



Corn

We should be exact about waiting here. A per-minute quota can recover after time passes. Calling every quota error permanent would be as misleading as calling every one temporary.



Herman

I was about to compress that too far. A minute allowance can replenish. A daily allowance needs a different timescale. An insufficient authentication scope won't improve with waiting at all. A project capacity problem may require configuration or a quota adjustment rather than another short retry.



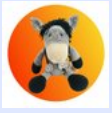
Corn

Sleeping doesn't grant permissions.



Herman

That's why the instruction to use exponential backoff is incomplete without error classification. Backoff helps when time or reduced pressure can change the outcome. It doesn't manufacture authorization or remove a hard ceiling.



Herman

People using rclone have reported excessive user rate limit errors during fast listing of shared drives, including cases where their visible project quota wasn't under pressure. That's a report of a troublesome pattern, not proof of exactly which hidden protection fired.



Corn

Fast listing seems an especially cruel label to leave enabled while nothing lists.



Herman

Another user reported that a sync completed only after reducing both transfers and metadata checkers to one apiece. The distinction matters. Download workers aren't the only source of traffic. Parallel checking can generate plenty of requests before the transfer display gets interesting.



Corn

I can already hear your objection to the apparent fix. One worker succeeding doesn't prove one worker is the universal answer.



Herman

You anticipated my objection before I could enjoy making it.



Corn

You've been looking at the word universal suspiciously all morning.



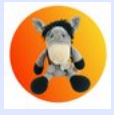
Herman

Different drives, projects, and operations have different bottlenecks. Still, those reports establish something useful. More concurrency can reduce useful throughput once it provokes rejection and retries.



Corn

The agent needs to count completed work, then. A terminal full of activity is a very poor substitute. I have seen your installations produce tremendous activity, Herman.



Herman

Some of that activity was diagnostic.



Corn

The backup remained a thought experiment.



Herman

That is why I now distinguish a successful command from a successful outcome. The Google Workspace CLI, gws, also deserves that distinction. Its own README says it is not an officially supported Google product, despite the name and its positioning for humans and agents.



Corn

An important sentence to read before delegating confidence.



Herman

It's pre-version-one software with a command surface generated from Google's Discovery Service. That helps it cover changing APIs, but it doesn't establish that every agent-generated workflow has sensible request economics. Dynamic command discovery and careful scheduling are separate achievements.



Corn

A tool can know every available operation and still choose an expensive way to do the job.



Herman

Now consider what happens when several layers all try to help. The library retries. The CLI retries. The agent sees no progress and reruns the CLI. A single intention has become overlapping attempts, possibly against the same shared allowance.



Corn

Persistence with multiplication.



Herman

There needs to be one coherent retry budget, or at least coordination between those layers. Otherwise the visible pause at one layer conceals pressure from another.



Corn

Here's a voicemail from a listener out there in the wild.
Tilda: Hey Corn, hey Herman, this is Tilda, I'm calling from the parking lot of the Piggly Wiggly on Highway Seventeen, I've got the truck running because it's already ninety-four degrees out here. I listen every week while I'm doing my grocery run, and today's episode about rate limiting and backup scripts got me thinking about something. I don't run any of that agent stuff myself, but my nephew set me up with one of those command line tools to back up my church choir photos to the cloud, and after about an hour it just quit on me and said I'd hit some kind of limit. So my question is, when it tells you that, is it telling you to wait a little while or to wait a long while? Because I waited twenty minutes like it said and it quit again right away, and I've got four hundred photos of the Easter cantata that still aren't backed up.



Herman

Tilda, that immediate second stop could mean the next run recreated the same burst, or that a different allowance hadn't recovered. The useful missing information is what limit was reached and whether the job resumed its progress. Elapsed waiting time alone doesn't tell you either.



Corn

Resuming matters enormously. If every restart repeats the inventory, the tool can spend the renewed allowance reaching the exact point where it stopped last time. Patient users become unpaid participants in a very slow loop.



Herman

Checkpointing changes that. Preserve completed work and the position in a long operation, where the API supports it. Then a pause can actually reduce outstanding work instead of resetting the task.



Corn

Also, there's a difference between retrying the rejected request and replaying the whole workflow. An agent saying I'll try again may mean either, and the user rarely gets asked which one they'd prefer.



Herman

Google recommends truncated exponential backoff for transient limits. The early waits are about one second, then two, then four, with a fresh random addition each time. The intervals grow rather than staying fixed.



Corn

A refusal gets progressively more personal space.



Herman

The random addition is up to a second in that example. Eventually the delay reaches a cap, typically thirty-two or sixty-four seconds. Truncated means it stops growing indefinitely, not that it stops retrying automatically.



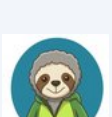
Corn

Which leaves the slightly awkward question of when it does stop.



Herman

A separate limit has to answer that. Maximum attempts, total elapsed time, or a job deadline. The documentation itself says clients should eventually be prevented from retrying indefinitely. Otherwise capped backoff can become a permanent background load.



Corn

I'd want an unattended job to stop with a useful account of what remains. Not silently surrender, and not spend three days interpreting no as encouragement.



Herman

You like unattended automation until it demonstrates your own resistance to getting up.



Corn

I stop when the job is impossible.



Herman

You also stop when the job is possible.



Corn

A broader compatibility range.



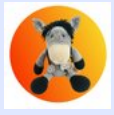
Herman

Jitter addresses a different problem from growing delays. Imagine many clients hit the same limit and all wait exactly the same interval. They return together, overwhelm it together, then retreat together. Randomizing the waits spreads the returning traffic.



Corn

A pause can be individually polite and collectively disastrous.



Herman

It can. There's another distinction I don't want to lose... backoff governs retries after trouble. Pacing governs normal traffic before trouble. A well-behaved client shouldn't require a stream of refusals to maintain a reasonable operating rate.



Corn

Reducing retries while keeping the same aggressive batch starts doesn't really learn anything. It just adds ceremonial waiting between collisions.



Herman

A feedback controller can lower concurrency or request rate when rejection rises, then cautiously explore more capacity. That is closer to discovering an acceptable rate than running the same burst forever.



Corn

Though we shouldn't casually retry every operation either. A failed response doesn't always mean the operation failed to happen. Reads are generally easier here. Repeating a write can create duplicate work unless the API offers a way to make retries safe.



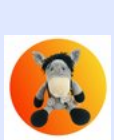
Herman

That distinction matters for agents doing more than backups. A request may have completed server-side before the connection failed. Retrying needs to account for operation semantics, not merely transport errors.



Corn

The model doesn't need to improvise all of this every time. I'd rather the tool own the scheduling and resumability, and the agent explain what the tool is doing. Otherwise a conversational interface becomes a new place to implement network congestion control badly.



Herman

Would better instructions from the server satisfy you, or would you still want an independent client-side cap?



Corn

Both. The server can describe its allowance, but my own cap can protect my bandwidth, budget, and other work. More importantly, the server's estimate might be stale by the time another worker spends the remaining capacity. An advertised allowance shouldn't be treated as a reservation.



Herman

The proposed RateLimit headers explicitly make that distinction. Available quota doesn't grant requests. Conditions can change, and other clients may share the same bucket.



Herman

There's active standards work at the IETF to give clients structured information before they repeatedly hit errors. The draft defines RateLimit-Policy and RateLimit fields for advertised policies and current limits. It remains a draft, rather than a promise that every API already speaks this language.



Corn

That would let the tool distinguish a replenishing allowance from a vague instruction to become less troublesome.



Herman

It can communicate the relevant partition, the effective window, and available quota. Retry-After can provide explicit pushback. The important change is that a client gets something it can schedule against, rather than inferring the rules entirely from rejection.



Corn

Provided the server actually publishes it.



Herman

Provided it publishes it, the client understands it, and both interpret the units consistently. A bare remaining balance means little if nobody knows whether the next operation costs one unit or two hundred.



Corn

The draft also separates reasons for trouble, doesn't it? That's the part I'd value most. I can tolerate a slower service much more easily than a service that won't say whether it thinks I'm unauthorized, expensive, or suspicious.



Herman

It defines problem types for quota exceeded, temporarily reduced capacity, and abnormal usage detected. Those convey meaningfully different situations. The last is much closer to an abuse-related signal than an ordinary exhausted allowance.



Corn

Finally, separate messages for the queue and the accusation.



Herman

Even then, abnormal usage doesn't establish malicious intent. It means the pattern triggered that category of protection. A legitimate application might need to change its behavior, while the operator may need to investigate a false positive.



Corn

Hm.



Corn

There's still a missing half. These headers tell my agent what the server wants. They don't tell the server that my agent belongs to somebody with a legitimate purpose. An attacker can read scheduling information too.



Herman

That is where Web Bot Auth enters. It uses HTTP Message Signatures to let automated clients cryptographically identify their requests. Instead of trusting a self-declared User-Agent name, the server verifies a signature against a public key associated with the agent.



Corn

Writing respectable crawler on the envelope has had a long run.



Herman

User-Agent strings are easily copied. IP verification has problems with shared hosting, VPNs, and privacy proxies. Per-site bearer credentials become cumbersome across large numbers of sites. Signatures offer a different basis for recognizing the sender.



Corn

Recognition still doesn't establish virtue. A signed agent can be signed and annoying.



Herman

Authentication identifies the signer. Authorization decides what that signer may do. Rate limiting still governs consumption. You can trust that a request came from a particular platform while refusing it because it exceeds capacity or violates your policy.



Herman

The signature carries a validity window and a key identifier. A Signature-Agent field points toward a directory where verification keys can be found. The signature covers the target authority, binding the proof to where the request is aimed. Those details make it stronger than a reusable label.



Corn

You're getting that look again. You like the cryptography.



Herman

I like replacing a guess with a verifiable statement.



Corn

Now verify that I get admitted.



Herman

That part remains policy. Cloudflare distinguishes signed agents directed by end users from conventional verified bots. Its initial examples include ChatGPT agent and Goose from Block, alongside browser infrastructure providers Browserbase and Anchor Browser.



Corn

Goose complicates the easy claim that everything here must run inside a giant platform. It's an open agent ecosystem. We shouldn't confuse who operates a particular request with the product name on the list.



Herman

Agreed... no, more specifically, recognition of Goose does not automatically mean every independent deployment has the same recognized identity. The protocol's ability to carry a signature and a site's willingness to trust that identity are separate questions.



Corn

Anyone can make a key. Not everyone can make a key anybody cares about.



Herman

That's the structural gap. An individual self-running a CLI can, in principle, implement signing. But that doesn't automatically earn directory inclusion, recognition by a security provider, or favorable treatment from websites. Cloudflare's enrollment involves an application process.



Corn

I'd call that a potential moat rather than a mathematical lockout. The cryptography isn't closed to individuals. The distribution of trust may be.



Herman

That is the distinction I would defend. A recognized platform identity can arrive with reputation and established relationships. A personal identity starts without those advantages. How much access it gets is an adoption and governance problem.



Corn

Your question, then. Would you route your personal backup through a recognized platform if that made it work reliably?



Herman

For some tasks, yes. For sensitive data, I'd want to understand the custody and permissions first. Reliability can tempt people into adding an intermediary they otherwise wouldn't choose.



Corn

That's my worry too. Self-hosting gives you control over your tool, but the website may know less about you than about the commercial intermediary. Privacy-preserving network choices can also make IP reputation less useful. The person trying to reduce disclosure can end up being asked to disclose more.



Herman

Google's experimental adoption shows the transition isn't complete even for major operators. It says only a subset of its agent requests are signed and tells site operators to fall back to IP-based verification.



Corn

New identity, old checkpoint.



Herman

Both approaches require participation on the other side. A client can't force an API to publish useful quota information, and a signature can't force a site to recognize its signer. Useful standards don't remove the defender's authority to say no.



Corn

Nor should they remove it. I'd settle for a no that stops wasting everybody's afternoon. A retry loop continuing after its user has given up isn't autonomy worth celebrating.



Hilbert

Yes, I agree there. I keep the rejected responses in a squared notebook. Just the time and the number. This month I was downloading drainage records from the council portal for my flat. Martin Bell in the records office said the export would be ready that afternoon. The link arrived three days later. My little downloader got a four twenty-nine and started again every five seconds. I had written that part myself.



Corn

Five seconds leaves plenty of room in the notebook.



Hilbert

I use narrow columns. Herman, your distinction between a pause and a restart was the part that mattered. Mine restarted the listing before it tried the download. Every attempt did the same preliminary work. I left it running while I went shopping. It had made no



Hilbert (continued)

progress when I got home. Take that chair scrape out before the word preliminary. The notebook is with Martin now, at the records counter. He asked for the error times. I wasn't copying them out twice.



Herman

Repeating the listing explains why the downloader could remain busy without getting closer to the file.



Hilbert

Five seconds seemed reasonable at the time. I wasn't asking for the whole borough. My brother-in-law maintains booking software and says people blame the limiter because the waiting is visible, while the repeated work isn't. He says the first retry often gets written carefully and the surrounding restart gets forgotten. He would say that. He's still trying to explain a booking system that sent me the same cancellation all afternoon. Martin eventually sent the records as an attachment. That took twelve minutes once he had my email in front of him. He had to check the drawings before he sent them.



Corn

...Whether an ordinary user can eventually get that kind of clear recovery without moving into a platform's identity system is what I'll be watching. Better quota disclosure and verifiable signatures can reduce guessing. They only broaden access if services actually support them and recognize identities beyond the established intermediaries.



Herman

Our thanks to Hilbert Flumingtop for producing the show.



Corn

If you want more of this, try episode five, Fine-Tuning ASR For Maximal Usability; episode nine, Benchmarking Custom ASR Tools - Beyond The WER; and episode sixteen, On Deepfakes, SynthID, And AI Watermarking. This has been My Weird Prompts. No question too obscure, no rabbit hole too deep.



Herman

If you enjoyed this one, leave us a review wherever you listen. It helps other curious people find the show.



Corn

See you tomorrow.

SOURCES

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

1. **IETF, 2026-05-23 . [primary]**
   https://www.ietf.org/archive/id/draft-ietf-httpapi-ratelimit-headers-11.txt

2. **IETF datatracker, updated 2026-05-23 . [primary]**
   https://datatracker.ietf.org/doc/draft-ietf-httpapi-ratelimit-headers/11/

3. **Google, limits updated 2026-05-01 . [primary]**
   https://developers.google.com/workspace/drive/api/guides/limits

4. **Google Crawling Infrastructure, experimental . [primary]**
   https://developers.google.com/crawling/docs/crawlers-fetchers/web-bot-auth

5. **Cloudflare blog, Forget IPs: using cryptography to verify bot and agent traffic . [primary]**
   https://blog.cloudflare.com/web-bot-auth/

6. **Cloudflare blog, The age of agents: cryptographically recognizing agent traffic . [primary]**
   https://blog.cloudflare.com/signed-agents/

7. **Cloudflare docs, last updated 2026-10-08 . [primary]**
   https://developers.cloudflare.com/bots/reference/bot-verification/web-bot-auth/

8. **IETF, Web Bot Auth architecture . [primary]**
   https://datatracker.ietf.org/doc/html/draft-meunier-web-bot-auth-architecture-04

9. **OpenAI Help Center, ChatGPT Work Cloud browser allowlisting . [primary]**
   https://help.openai.com/en/articles/11845367-chatgpt-agent-allowlisting

10. **`gws` repo, last push 2026-10-26, v0.22.5 . [primary]**
    https://github.com/googleworkspace/cli

11. **Wikipedia, last edited 2026-09-02.**
    https://en.wikipedia.org/wiki/Rate_limiting

12. **Wikipedia, last edited 2026-10-05.**
    https://en.wikipedia.org/wiki/Exponential_backoff

13. **rclone forum.**
https://forum.rclone.org/t/google-drive-only-works-with-transfers-1-and-checkers-1/31517

14. **rclone forum.**
https://forum.rclone.org/t/userratelimitexceeded-while-using-fast-list-on-shared-google-drive/29986

15. **rclone forum.**
https://forum.rclone.org/t/googleapi-error-403-user-rate-limit-exceeded-rate-of-requests-for-user-exceed-configured-project-quota/22396

16. **rclone issue #3857.**
https://github.com/rclone/rclone/issues/3857

17. **Google Developer forums.**
https://discuss.google.dev/t/i-need-help-with-the-quotas/324042/7

18. **third-party `gws` troubleshooting notes.**
https://github.com/alirezarezvani/claude-skills/blob/main/engineering-team/google-workspace-cli/skills/google-workspace-cli/references/troubleshooting.md

19. **Castle blog.**
https://blog.castle.io/how-to-authenticate-openai-operator-requests-using-http-message-signatures/