Episode Book

EPISODE #6394

Why API Email Still Relies on SMTP

PUBLISHED	31 October 2026 at 20:17 UTC
DURATION	20:36
LISTEN	myweirdprompts.com/episode/api-email-smtp-infrastructure-53080/
HOSTS	Corn, Herman, Larry, Hilbert
SCRIPT MODEL	anthropic/claude-sonnet-5.5
VOICES	chatterbox-regular (chatterbox-local)
PIPELINE	V5.26
SOURCES	25 cited by the research (listed at the end)
BOOK GENERATED	1 November 2026 at 07:41 UTC

SYNOPSIS

Developer email services like Resend seem to replace decades-old protocols with a simple HTTPS POST, but the transport layer remains entirely intact underneath. Modern API mail platforms handle queues, connection pools, and MTA operations like bounce classification and retries behind the curtain. While open-source projects now build personal inboxes on top of these APIs, DNS setup, port blocks, and inbox archiving reveal why transactional mail interfaces still struggle to become everyday mailboxes.

TRANSCRIPT



Corn

A man named Gerald, apparently a donkey, knows a great deal about routing. I'll come back to that. Daniel's question is about email, and why the one piece of internet plumbing nobody loves never got its API moment.



Corn

His setup is this. Regular people are stuck with SMTP and IMAP, a server holds your mail and your client connects to it. Programmatic email over an API has always been a developer thing for transactional mail. Then Resend made it painless, and someone built a personal client on top of it, which Daniel takes as proof it needn't be enterprise-only. So what does API email look like as an architecture, how much SMTP and IMAP is still down there, and does any of it fit how people actually use Gmail?



Herman

Herman Poppleberry, reporting for the SMTP desk. Start with what happens when you hit send. Your code makes an HTTPS POST to Resend's API. A control plane validates it. Resend splits its stack into a control plane that handles requests and a data plane that sends the email, with queues between them to absorb spikes and keep the services separate.



Corn

Why the queues? Couldn't the control plane just hand the message straight over?



Herman

It could, until a customer fires off a hundred thousand password resets at once. Without a queue, the sending side gets hammered directly and the API starts timing out. With a queue, the API accepts the request quickly, the message waits its turn, and a slow receiving provider can't back up into the front door. It's the same reason a restaurant has a ticket rail between the host stand and the kitchen.



Corn

So the API can say "got it" long before the email has actually gone anywhere.



Herman

Yes, and that's a deliberate design choice. The acknowledgment means accepted, not delivered. Keep that distinction in mind, because it comes back.



Herman

From the queue, the data plane hands the message to an MTA, and the MTA talks to the recipient's MX server. So the API is the front door. The transport layer is all still there behind it.



Corn

Then it's not really a replacement for anything.



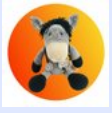
Herman

It's an interface. And Resend proves it by offering a second front door to the same backend. There's an SMTP endpoint, smtp.resend.com, on ports twenty-five, four sixty-five, five eighty-seven, and a couple of 2000-series ones. The username is literally "resend" and the password is your API key.



Corn

So the API key is the password. Charming.



Herman

Two interfaces, one delivery engine. Which answers the question of whether it skips the MTA. It doesn't skip it. It is the MTA, just not yours.



Corn

Does Resend run its own, though? I assumed everyone in this space was renting someone else's.



Herman

They were, until recently. Resend described running on a rented foundation, then built Forge, their own MTA, delivering from IP blocks they own and operate. And Forge still speaks plain SMTP to the outside world. Their MTA engineering job posting reads like a table of contents for everything the API hides: retries, backoff, throttling by domain, by provider, by customer, by IP. Shared pools, dedicated IPs, warmup, reputation-aware routing, bounce classification into hard, soft and deferral.



Corn

What's the practical difference?



Herman

A hard bounce is permanent: the mailbox doesn't exist, stop trying, and stop sending to that address or your reputation suffers. A soft bounce is a maybe, like a full mailbox. A deferral is the receiver saying not right now. The classification matters because the right response differs. One gets suppressed, one gets retried, one gets slowed down.



Corn

And if you get it wrong? Say you treat a deferral like a hard bounce.



Herman

Then you suppress a perfectly good address and a customer never receives their receipt. Go the other way, and you keep hammering a dead mailbox, which is exactly what spam filters read as sloppy list hygiene. Both mistakes cost you, just in different currencies.



Corn

That's a lot of work for a company that started by saying the incumbents were too enterprise.



Herman

We'll get there. Take retries first. A receiving server says try again later, which is greylisting or a temporary rate limit. Forge keeps trying for up to forty-eight hours. Other temporary failures fall back to the older engine, which tries for up to fourteen hours. After that the message is marked bounced. If Forge has an incident, delivery drops back to the legacy engine automatically.



Corn

So "sent" in your code means "queued, and someone else will agonize over it for two days."



Herman

Correct. And your code sees none of that agony. Activation is the same story. Turning Forge on needs one CNAME record for the sending domain, the Return-Path subdomain, which receivers use for bounces and SPF alignment, plus an MX update for receiving. No code changes. DNS changes.



Corn

"No code changes" is where the work moves into DNS.



Herman

Right. Now efficiency, because people assume the gain is skipping the handshake. The best hard number I know is from Mailgun, back in twenty nineteen. Their old engine opened a fresh connection per message. TCP handshake, SMTP handshake, and over TLS a TLS handshake on top. It could take over a minute and a half to deliver one message.



Corn

A minute and a half. I've had naps shorter than that.



Herman

Then they pooled connections and sent many messages per connection, and what took ninety seconds took under roughly six hundred milliseconds. So the gain isn't skipping SMTP. It's paying the handshake once and amortizing it across a pool of warm connections.



Corn

Which only works if you have enough volume to keep the pool warm.



Herman

That's the point. A person self-hosting sends a few messages a day. They get none of that benefit. Concentration is the efficiency. It's like a bus versus a taxi. One rider per trip is expensive, and a full bus on a fixed route is cheap.



Corn

Does that apply to the serverless crowd too? Daniel mentioned developers.



Herman

That's a separate argument, from Vercel. An SMTP send depends on one stateful connection. DNS lookup, TCP, TLS, auth, transfer, disconnect, each step needing the last on the same socket. An HTTP API send is one stateless request, one round trip. And inside a serverless function it gets nasty. If you forget an await, the function returns before the SMTP exchange finishes and the connection drops. Nothing throws, so your logs stay clean while the message never leaves.



Corn

That's my favorite kind of bug. The kind that reports itself fine.



Herman

Which is a clinician's nightmare, by the way. Also, Vercel blocks outgoing port twenty-five, though four sixty-five and five eighty-seven stay open. And I'll flag that the often-quoted two hundred to a thousand milliseconds of SMTP overhead per send comes from vendor blogs. Treat it as a guess.



Corn

Why would a serverless function make that worse than a normal server?



Herman

Because a normal server stays alive. It can hold a connection open, finish the conversation in the background, and retry. A serverless function is frozen or killed the moment it returns a response. A long conversation that needs the process to live on is exactly what that model punishes, while a single HTTP request that completes and returns fits it perfectly.



Corn

So the failure isn't SMTP being bad. It's the function's lifetime not matching the conversation's lifetime.



Herman

That's a cleaner way to say it, yes. Think of leaving a phone call mid-sentence because your shift ended. The other end is still talking, and you're gone.



Corn

So the speed story is mostly "someone big does the boring part for you."



Herman

Mostly, yes.



Corn

So the question is whether API-based email can ever work for regular people who just want a mailbox, not a developer console.

L

Larry

Folks, are you tired of your email being held hostage by servers and protocols you do not understand? Then you need Personal Postman, the world's first fully API-based mailbox that skips SMTP, skips IMAP, and skips straight to the good part, which is me reading this advertisement. It is a little brass box that sits on your desk and accepts a single HTTPS POST from anyone you love. The price was one hundred and twenty dollars, but because I am still talking it is now eighty dollars, and now fifty five dollars, and now thirty dollars, and I will also throw in a spare front door, a control plane you can use as a coaster, and a small donkey named Gerald who knows a great deal about routing. Wait, no, the deal just ended. The deal ended before I finished describing it. That is how limited it was.



Herman

Skips SMTP. It does not skip SMTP. I just spent six minutes on this. And a control plane is not a coaster, it's the part that validates requests, and you cannot hold it, it's a service on Fargate.



Corn

Let it go, Herman.



Herman

And a brass box accepting a POST from anyone you love would need a public endpoint and authentication, or your mother is posting into your house.



Corn

Herman. Gerald is fine.



Herman

Gerald is not a router. Fine. Moving on. Personal adoption.



Corn

Yes. There's a real cluster of personal clients on Resend, right?



Herman

A whole cluster, mostly open source. Inboxes is the one that's most polished: bring your own API key, threads, full-text search, snooze, scheduled send up to thirty days out, undo send with a five, ten or thirty second hold. It scores spam from SPF, DKIM and DMARC. It even has an MCP server so a coding agent can read and draft mail, with agent sending off by default and enforced server-side. Free self-hosted, five dollars a month hosted per organization.



Corn

Off by default is the right setting for an agent with my inbox.



Herman

Enforced server-side is the important part. A setting in the client is a suggestion, because a misbehaving agent can ignore it. If the server refuses the send, it doesn't matter what the agent decides to try.



Corn

How does undo send work, if the message is already on its way? That always seemed like magic.



Herman

It isn't magic, it's a delay. The message sits in a holding state for five, ten or thirty seconds before it's handed off. Undo just cancels it while it's still waiting. Once it's gone to the MTA, nobody can recall it.



Herman

There are others. Vemail ships as one Docker image on a self-hosted Convex backend. Mailflow is a Next.js inbox. Sendbox gives you unlimited identities with Postgres, receiving through Resend's inbound webhooks. Emailer sits on top of several Resend accounts and encrypts the keys.



Corn

Why do all of them exist? Resend sends mail. Why can't it also hold it?



Herman

That's the gap. Resend sends your email but has nowhere to read the replies, and stored mail gets deleted after days. Its own docs frame the product as transactional email and marketing broadcasts, not personal correspondence. So every one of these projects is filling an inbox-shaped hole.



Corn

Then what actually stops a normal person? Is it the protocol?



Herman

No. First, you need your own domain, verified with Resend, plus an API key. Second, it's a sending platform, so you run or pay for a separate archive. Third, DNS literacy. Choosing a Return-Path subdomain, getting SPF alignment right, CNAME conflicts, seventy-two hours of propagation, and if you're on Cloudflare the record must be DNS only, not proxied.



Corn

Seventy-two hours to find out you typed one letter wrong.



Herman

And the proxy setting is a trap because it's the default behavior for web records. Cloudflare's orange cloud routes traffic through its network, which is great for a website and wrong for a mail record. Receivers need to see the real target, so the proxied record quietly breaks the thing and nothing tells you why.



Herman

Fourth, ports. The author of a self-hosted gateway called Posthorn found DigitalOcean and other VPS providers blocking default SMTP ports, so he ended up with a Postfix relay hack. And fifth, Forge itself only runs in us-east-1, has no open or click tracking, and can't be used on dedicated IP domains.



Corn

So the barrier is onboarding. Nobody wants to be their own postmaster.



Herman

Right, and here's the irony I like. Resend's founder pitched the company against SendGrid, Mailgun, Postmark and SparkPost, all founded around two thousand nine or ten and all acquired, saying nobody was building for developers. And Resend's biggest move since is owning IP blocks, warmup, reputation routing, postmaster relationships. That is the most enterprise-shaped work in email.



Corn

Developer-first meets deliverability. Deliverability wins.



Herman

Warmup is a good example. A brand-new IP has no reputation, so receivers treat it with suspicion. You send small volumes, increase gradually, and prove you're not a spammer. You can't shortcut it, which is why owning clean IP blocks is such a moat.



Corn

So a new sender is a stranger knocking on a door at midnight.



Herman

And a warmed IP is the neighbor who's waved over the fence for years. Same knock, very different reception.



Herman

Now IMAP. Daniel asks if it was built for a world that no longer exists. I'd say it was built for exactly this one. The spec says it was designed to permit complete management of a mailbox by multiple clients, leaving messages on the server until you delete them. That's webmail. Server-authoritative, many devices.



Corn

So Gmail is an IMAP server with a website?



Herman

Gmail's web client and its IMAP endpoint are two front ends on one store. Same pattern as Resend's API plus SMTP. So what's actually being displaced is two legs. The client to server leg, IMAP and POP3, and the app to relay leg, SMTP submission. HTTPS plus webhooks replace both. SMTP between domains stays universal.



Corn

Contrast that with POP3, though. That one assumed something different.



Herman

POP3 assumed one computer. Download the mail, delete it from the server, done. That works if you have a single desktop. IMAP came along because people had a laptop and a work machine and wanted the same inbox on both.



Corn

Then the stateful socket is the legacy part, not the protocol.



Herman

Yes. And programmatic email suits multi-client mail better than a long-lived socket does. The architecture fits. The onboarding doesn't. Last bit: infrastructure is closing the SMTP door itself. Vercel blocks port twenty-five, VPS providers block defaults. The shift may come less from elegance than from platforms making the old path impossible.



Corn

Forced migration. The most reliable kind.



Hilbert

...and it was the CNAME on the Return-Path subdomain, Corn. Sat there the whole of a weekend. Cloudflare had it set to proxied. Orange cloud. It wants grey. DNS only. I had a customer, a Mrs. Okonkwo, who runs a bakery supply shop, and her invoices were going to junk at two providers, and she rang me every few hours that Saturday.



Hilbert

The record was fine. Seventy-two hours they tell you for propagation. Nobody says that most of it is people not checking. I printed the dashboard screen and taped it inside the cabinet over the kettle, with the CNAME highlighted in yellow. Highlighter was a Stabilo.



Herman

That's actually a sensible place for it. Where's the bounce history from? You'd have to log those somewhere.



Hilbert

Notebook. Every bounce on my domain since twenty nineteen, sorted by receiving provider. Greylisting is a personality trait. Some servers just do it. Dennis Farrow ran a mail room I worked with and could clear four hundred and twelve messages an hour. Nobody has beaten it. I still know that number.



Herman

Sorted by provider is smart. Patterns show up that way. If one provider defers everything from a new sender, you see it at a glance instead of guessing.



Hilbert

Exactly the use. Three providers do nothing but defer the first attempt. After that, they're fine.



Herman

That's greylisting doing its job. The first attempt gets a temporary refusal, and a legitimate server retries while a lot of spam software never bothers. So the retry window Forge keeps is what gets you through.



Corn

The notebook is arguably the only complete bounce archive Resend's seven-day retention can't touch.



Hilbert

It's three notebooks now. The first one went to Mrs. Okonkwo's nephew, who needed something to write his lines in. Fine, Corn. It's not a database.



Corn

It's a database that takes up a shelf.



Herman

A shelf is a data plane.



Corn

Don't.



Corn

If the API is just a front door onto the same SMTP engine, what actually changed? Maybe only who owns the MTA and who has to think about DNS.



Herman

And whether someone builds a consumer front end that hides the DNS. The inbox projects exist and the demand is there. Until the CNAME and the proxy toggle disappear behind a signup screen, it's a personal tool for people who own domains and tolerate orange clouds. A seventy-two hour wait with a bakery supply customer on the phone doesn't suit most people.



Corn

So the model is closer to how Gmail hides everything. You sign up, and you never learn what an MX record is.



Herman

Gmail's real product was never the protocol. It was that nobody had to be a postmaster. Whoever does that for an API-first mailbox wins the personal market.



Corn

Thanks to Hilbert Flumingtop for producing. If this was your kind of episode, go back for episode thirty-five, The Privacy Gap; and episode ninety-one, The Story Behind the Show. This has been My Weird Prompts. You can find every episode at my weird prompts dot com.



Herman

We'll be back soon.

SOURCES

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

1. **Resend docs, Introduction [primary]**
   https://resend.com/docs/introduction
2. **Resend docs, Send emails with SMTP [primary]**
   https://resend.com/docs/send-with-smtp
3. **Resend engineering handbook, What is our tech stack [primary]**
   https://resend.com/handbook/engineering/what-is-our-tech-stack
4. **Resend blog, Announcing Resend Forge [primary]**
   https://resend.com/blog/announcing-resend-forge
5. **Resend docs, Resend Forge [primary]**
   https://resend.com/docs/resend-forge
6. **Resend, Resend Forge feature page [primary]**
   https://resend.com/features/forge
7. **Resend, SMTP Service feature page [primary]**
   https://resend.com/features/smtp-service
8. **Resend docs, Receiving emails through Resend [primary]**
   https://resend.com/docs/dashboard/receiving/introduction
9. **Resend job posting, Backend Engineer, MTA [primary]**
   https://jobs.ashbyhq.com/resend/053bde8f-294e-4cce-9d62-2301282120a2
10. **Hacker News, Launch HN: Resend (YC W23), 2023-06-13**
    https://news.ycombinator.com/item?id=36309120
11. **Hacker News, Show HN: Posthorn, self-hosted mail gateway, 2026-05-27**
    https://news.ycombinator.com/item?id=48289624

12. **Mailgun blog, 2019-07-16**
https://www.mailgun.com/blog/it-and-engineering/how-connection-pooling-helped-us-cut-delivery-time-in-half-offer-opportunistic-tls/

13. **Vercel KB, published 2026-09-09, updated 2026-09-24**
https://vercel.com/kb/guide/sending-emails-from-an-application-on-vercel

14. **Wikipedia, Internet Message Access Protocol, last edited 2026-09-17**
https://en.wikipedia.org/wiki/Internet_Message_Access_Protocol

15. **inboxes, The Missing Inbox for Resend**
https://inboxes.net/

16. **Inboxes, Webmail & Inbox for Mailgun and Resend**
https://inboxes.pro/

17. **inboxes docs, Getting Started**
https://inboxes.net/docs

18. **vemail README**
https://github.com/sayedhfatimi/vemail

19. **Mailflow README**
https://github.com/Flowdesktech/mailflow

20. **Inboxes README**
https://github.com/inboxesnet/inboxes

21. **Emailer project page**
https://joshbaker.gg/projects/emailer

22. **sendbox README**
http://github.com/coreyepstein/sendbox

23. **Resend Mail Client README**
https://github.com/IgorShadurin/resend-mail-client

24. **Simple Email Client README**
https://github.com/ProjexaAI/Email-Client

25. **Autosend, SMTP vs API: Complete Developer Guide (vendor estimate)**
https://autosend.com/blog/smtp-vs-api-complete-developer-guide