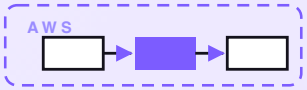
7-PART SERIES · FREE COMPANION



Email assistant

A serverless email assistant on AWS that triages your inbox, drafts replies from your own knowledge, and escalates anything beyond its remit to a human. Seven posts on the same system — one diagram at a time — with an engineering reference at the end.

CONTENTS

Email assistant

PART 1 OF 7

APRIL 29, 2026 PART 1 OF 7 · EMAIL ASSISTANT SERIES ~5 MIN READ

An email assistant on AWS for a few dollars a month

Your inbox is doing too much work. Most of what lands there is repeats — the same five questions, asked in different words. Here's how to build a small email assistant that reads each new message, replies from your own knowledge when it can, drafts the rest for your approval, and quietly escalates anything important.

KEY TAKEAWAYS

- Three outside surfaces, three AWS pieces. Reader, brain, and sender turn one inbox into one of four moves per email.
- Four moves, always: *answer directly*, *draft for review*, *escalate to a human*, *archive without replying*. Strict tool_use makes a fifth impossible.
- The brain answers from one Drive doc — FAQs, hours, prices, tone, sender allowlist. It never invents prices, hours, or promises.
- Auto-send only above a confidence bar. Borderline confidence becomes a draft you approve in seconds; below it, the email escalates.
- Runs on AWS for about $3/month at typical small-business volume — SES inbound, Lambda, Bedrock Haiku, plus an S3 Vectors index over your knowledge file.

The whole system on one page

Before any code, here's the shape of what we're building.

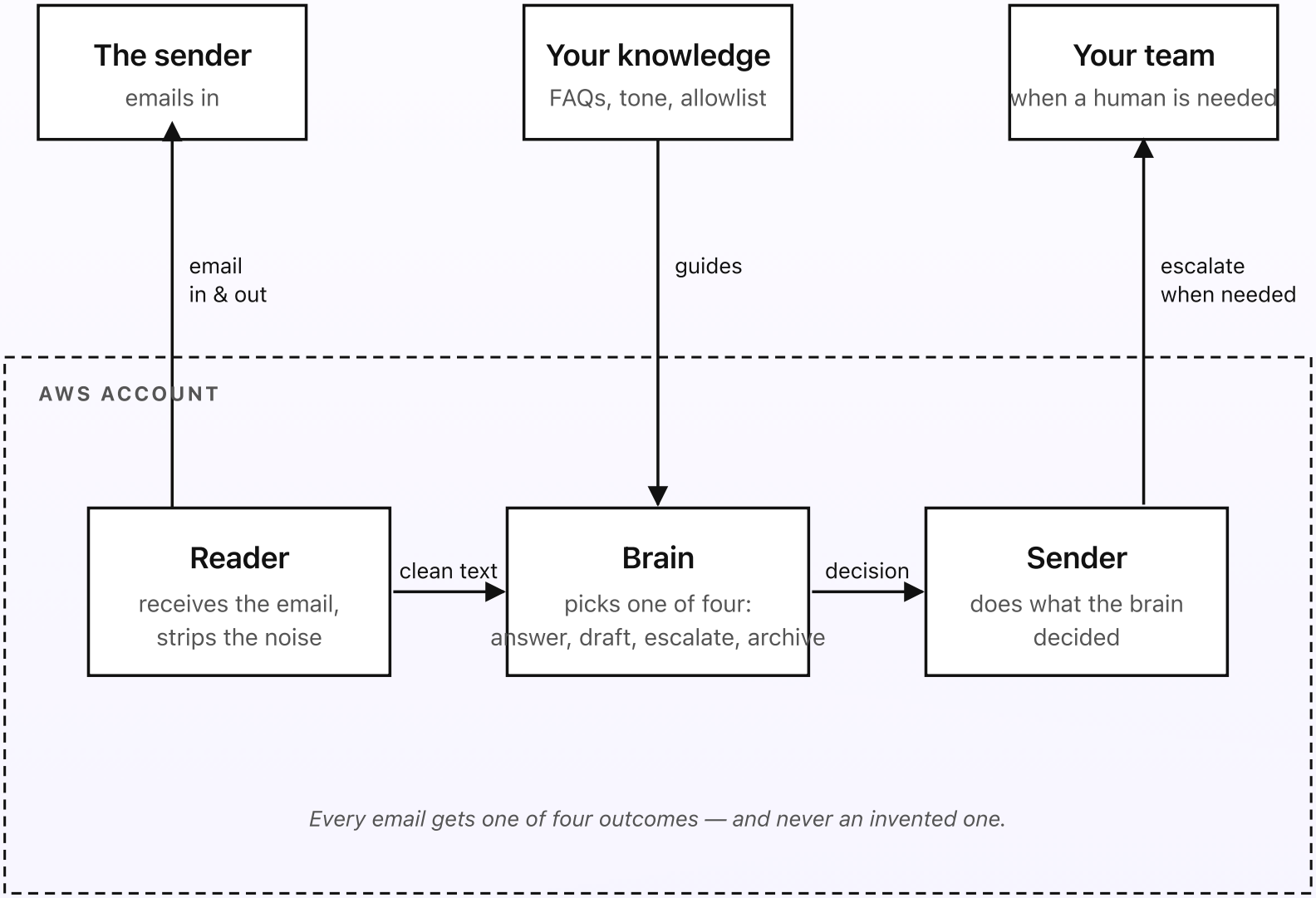

The sender
emails in
Your knowledge
FAQs, tone, allowlist
Your team
when a human is needed
email
in & out
guides
escalate
when needed
AWS ACCOUNT
Reader
receives the email,
strips the noise
clean text
Brain
picks one of four:
answer, draft, escalate, archive
decision
Sender
does what the brain
decided
Every email gets one of four outcomes — and never an invented one.

Fig 1. Three outside surfaces, three pieces inside AWS. Mail in, reply or escalate back, with a four-tool brain in the middle.

What you set up once (the outside)

- **A receiving address** — `hello@yourbusiness.com` or similar. Either pointed straight at the assistant, or forwarded from your existing inbox so nothing about your address changes for senders.
- **A short knowledge file** — your FAQs, your hours, your prices and policies, the tone you want, and a small allowlist of senders who should always go straight to a human. Lives in a Google Doc you can edit anytime.
- **An escalation inbox** — the address the assistant forwards to when an email needs a human. Your normal inbox, a shared team inbox, a help-desk queue — whichever you already use.

What runs on every email (the inside)

- **The reader** — takes the raw inbound email and turns it into a clean, focused message. Strips quoted threads, signature blocks, footers, and trackers, so the brain reads what the sender actually meant to say.
- **The brain** — reads the cleaned email and picks one of four moves: answer directly from the knowledge file, draft a reply for your review, escalate to a human, or archive without replying.
- **The sender** — carries out the decision. Sends the auto-reply with proper threading. Or queues a draft you approve in seconds. Or forwards the email plus a one-line summary to your team. Or files it quietly.

In plain words

An email lands. The cloud reads it. A small AI decides whether it's a question your knowledge file can answer, a question that needs a human, or noise. If it can answer, it does — in your tone, citing your own words. If it can't, it puts a draft on your desk or hands the whole thing to your team. The sender hears back fast, and you stop spending mornings retyping the same five replies.

Total cost runs in coffee-money territory at typical small-business volume — a few cents per email, going up smoothly with how often the inbox rings.

DESIGN RULES THAT SHAPED EVERY DECISION

- Stay inside the AWS always-free quotas where possible. SES has a per-email cost, but it's in fractions of a cent.
- The assistant answers from your knowledge file only — never invents prices, hours, or promises.
- Auto-send only when confidence is high. Anything borderline becomes a draft you approve, not an apology you send later.
- Threading must be preserved. The reply must show up under the original message in the sender's mail client, not as a fresh thread.
- Configuration lives in a Drive doc you can edit. Updating tone, hours, or the allowlist never needs a deploy.

Why this shape

Most “AI for email” tools fall over for one of three reasons. They charge a per-seat fee that costs more than the work they’re doing. They make up prices and policies in their replies. Or they’re a black box where your customer history sits inside someone else’s product.

The setup above is the smallest one I could find that handles all three. One way in (your address), one way out (your team or the sender), three small pieces in the middle that read, decide, and act. Everything stays in your AWS account, so the data is yours and you pay per use, not per seat.

The next five posts walk through each piece in turn — how an email enters and gets routed, how the reader strips it down, how the brain picks one of four moves, how a reply stays accurate, and what the whole thing actually costs. One diagram per post. A final engineering reference at the end gives engineers the dense version with precise service names and model IDs.

PART 2 OF 7

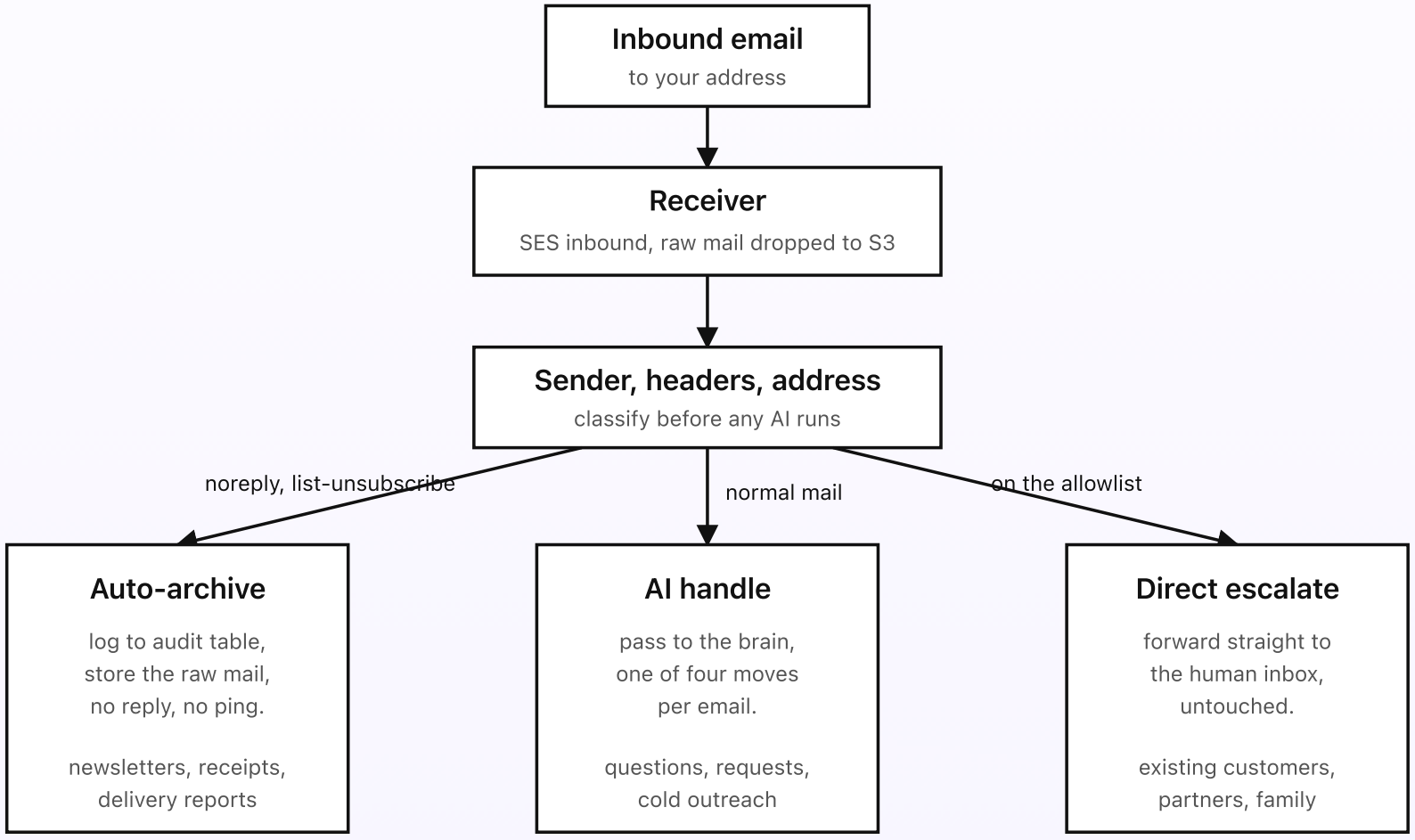
APRIL 29, 2026 PART 2 OF 7 · EMAIL ASSISTANT SERIES ~4 MIN READ

How an email enters the assistant

Not every email deserves the brain. Newsletters, “your package shipped” receipts, and DMARC reports should be filed silently. VIPs should reach you without an AI in the way. Everything else gets the assistant. Three lanes at the door.

KEY TAKEAWAYS

- Three lanes at the door — *auto-archive*, *AI-handle*, *direct escalate* — and only AI-handle ever reaches the brain.
- Auto-archive matches three signals: `noreply@` / `no-reply@` / `donotreply@` / `mailer-daemon@` senders, `List-Unsubscribe` headers, and DMARC or bounce reports.
- Direct escalate runs from a short allowlist — key clients, active partners, your own staff, replies to threads a human already started — and forwards untouched with an `X-Assistant-Lane: direct` header.
- Cheap rules at the door save real money: a typical SMB inbox is half noise, and routing it past the model before any token spend cuts cost five-fold.
- Every inbound message lands in S3 first; the AI-handle lane writes a parsed envelope to DynamoDB `tbl-messages` so the eventual reply still threads correctly weeks later.



By the time the brain runs, the noise is gone and the VIPs are already through.

Fig 2. Three lanes at the door. The brain only sees mail that's actually for it.

Why route before you read

It's tempting to send every email to the brain and let the AI sort it out. Don't. Two reasons.

First, cost. A typical small-business inbox is half noise — receipts, newsletters, calendar invites, unsubscribe-style mass mail, vendor pings. Running each of those through a language model is paying for thinking you don't need. Cheap rules at the door drop most of it before any AI runs.

Second, trust. Some senders should never be answered by an AI. A long-time customer asking about their order, a partner pinging you about a contract, a friend emailing the business address — if the assistant replies to those, you've damaged a relationship to save a minute. The allowlist sends those straight to your inbox.

Lane 1 — Auto-archive

The first thing the receiver checks is whether this email even wants a reply. Three signals usually answer that:

- **The sender is a noreply address.** `noreply@`, `no-reply@`, `donotreply@`, `mailer-daemon@` — the sender is telling you, in their address, not to write back.
- **The email has a `List-Unsubscribe` header.** A standards-compliant signal that this is bulk mail, not personal correspondence.

- **The email is a delivery or auth report.** DMARC reports, bounce notifications, calendar invites with auto-handle headers.

If any of those match, the email goes to the auto-archive lane. The raw message is stored in S3, a row is written to the audit table, and the lane ends. No reply, no notification. If you ever need to look back — "did the package-tracking email come in?" — you can; the data is there. But it doesn't live in your inbox.

Lane 2 — AI handle

The default. Anything that isn't obviously noise and isn't obviously a VIP goes to the brain. This is where the work happens, and it's the subject of the next three posts: how the reader cleans the message, how the brain picks one of four moves, and how a reply stays accurate.

The receiver does one more thing before passing the email through: it stores the raw mail in S3 and writes a small "message envelope" record — sender, subject, timestamp, message ID, threading headers — into a small DynamoDB table. The brain reads from that envelope; the raw mail stays in S3. This separation matters when the assistant has to send a reply that threads correctly, weeks later, against an old message.

Lane 3 — Direct escalate

The allowlist. A short list of email addresses, domains, or patterns that always skip the AI. Examples that usually belong on it:

- The personal addresses of your most important clients.

- Domains of partners or vendors you're actively working with.
- Anything from your own staff's personal addresses.
- Anything that arrives in reply to a thread a human already started.

For senders on the list, the receiver forwards the message to your normal inbox unchanged, with a small header tag (`X-Assistant-Lane: direct`) so you know it bypassed the AI. The thread stays intact, the sender notices nothing, and you stay in the loop on the relationships that matter most.

In plain words

Most of what hits a business inbox doesn't need an AI to handle, and some of it shouldn't. Three short rules at the door — archive the obvious noise, escalate the people you always want to see, send the rest to the brain — mean the AI only ever runs on what it's actually good at.

PART 3 OF 7

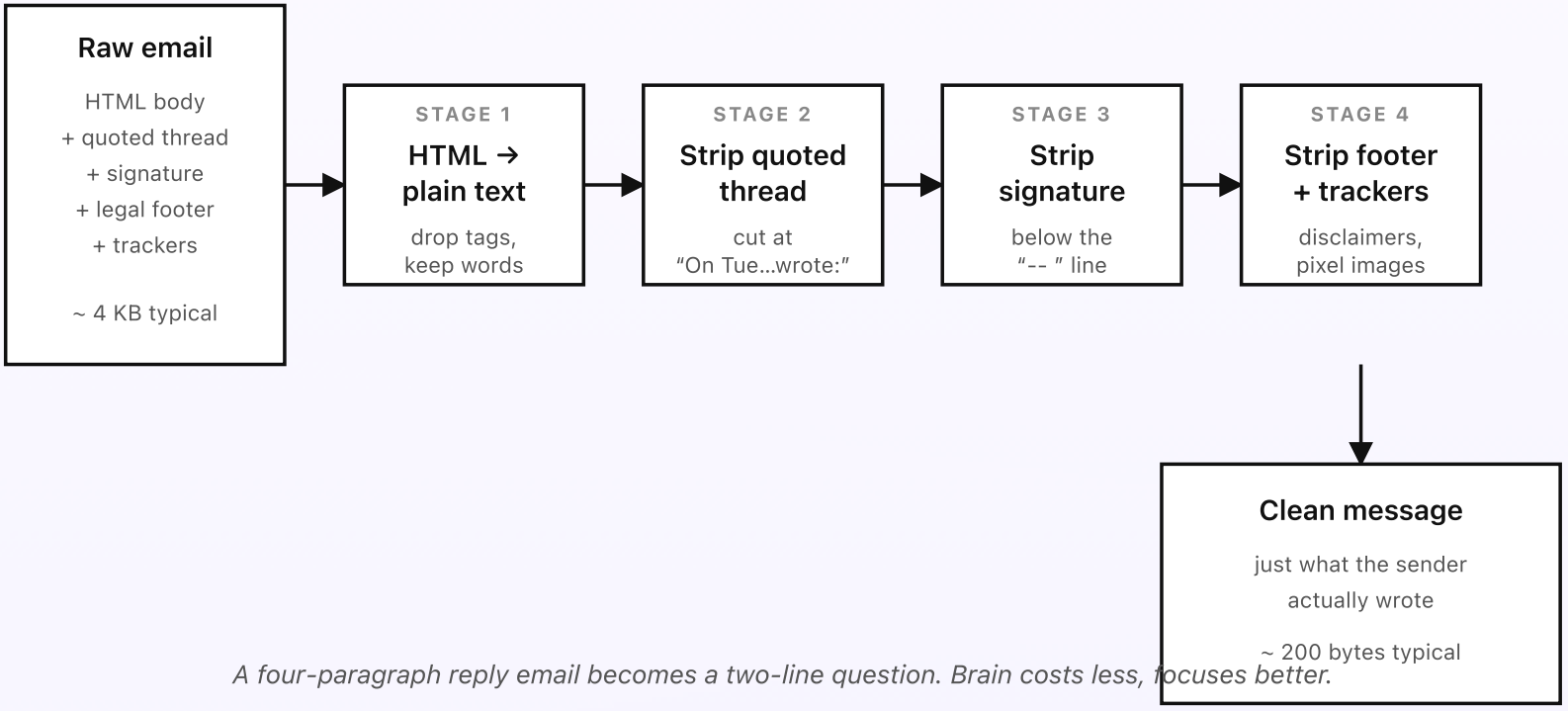
APRIL 29, 2026 PART 3 OF 7 · EMAIL ASSISTANT SERIES ~5 MIN READ

How the assistant reads an email

A real email is mostly not the message. It's a quoted thread, a signature, a legal footer, a tracking pixel, and one short paragraph of actual question. The reader's only job is to find that paragraph.

KEY TAKEAWAYS

- Four stages run in plain code before any model token is spent: HTML-to-text, strip the quoted thread, strip the signature, strip footers and trackers.
- Quoted-thread cuts trigger on four predictable patterns: Gmail's `On <date>, <name> wrote:`, Outlook's `From: … Sent: …`, lines of `>` at column zero, and legacy `--- Original Message ---`.
- Signatures cut on the RFC `-- ` line first, then heuristic closings ("Best,", "Thanks,") — the sender's name and contact details survive on the message envelope, just not in the body the brain reads.
- A typical reply is 90% quoted history and 10% new content; stripping the thread cuts model input five to ten times and the bill drops with it.
- Cleanup is conservative: the original raw MIME stays in S3 alongside the cleaned body, so any wrong reply can be traced back to what the sender actually wrote.



A four-paragraph reply email becomes a two-line question. Brain costs less, focuses better.

Fig 3. Four small steps. The brain only ever sees the last box.

Why bother with parsing?

Three reasons. The first is cost. The brain charges per word it reads. A typical reply email is 90% quoted history and 10% new content. If you send the whole thing to the model, you pay to re-read what you already saw. Stripping the quoted thread alone usually cuts model input by five to ten times.

The second is accuracy. Models get confused when half the input is the previous reply quoted back at them. They start summarising the thread instead of answering the new question. Feed them just the new content and they answer the new question.

The third is privacy. The thread might carry personal details the original sender never expected an AI to read — old phone numbers, billing info, screenshots. Stripping it before any model sees it is the cleanest way to respect that.

Stage 1 — HTML to plain text

Most email arrives as HTML even when it looks like plain text. Inline styles, table layouts, image links, conditional Outlook markup. The first stage drops every tag and keeps the text content, normalising whitespace and resolving common entities.

The output looks like what you'd see if you copied the email body and pasted it into a notes app. No formatting, no images, just words.

Stage 2 — Strip the quoted thread

Email clients all use slightly different quoting conventions, but the patterns are predictable. The reader cuts the message at the first occurrence of any of these:

- `On <date>, <name> wrote:` — Gmail, most modern clients.
- `From: <name> Sent: <date>` — Outlook's header-style quoting.
- A line of `>` characters at column zero — older clients, plain-text mailing lists.
- `--- Original Message ---` — legacy email forwarding.

Whatever sits above the cut is the new content. Whatever sits below it is history that's already in the audit log if anyone needs it.

Stage 3 — Strip the signature

Signatures are noisier than they look. "Sent from my iPhone" counts. So does the four-line title-and-phone-number block, the marketing tagline, the social media icons, the "think before you print" environmental footer.

The convention is a line containing exactly `-- ` (two dashes and a space) above the signature. When that signal isn't there, the reader uses heuristics: lines that look like contact details, lines that match common closings ("Best," "Thanks," "Regards," the sender's name on its own line), short last paragraphs that look like a sign-off.

A note: the reader keeps the sender's name and contact details in a separate field on the message envelope — just removed from the body the brain reads. The

brain doesn't need the signature to know who sent the email; the envelope already says so.

Stage 4 — Strip footers and trackers

Bulk-mail and corporate footers ("CONFIDENTIALITY NOTICE", unsubscribe links, image trackers, marketing disclaimers) get cut last. These are usually long, never relevant to the message, and almost always anchored on phrases like:

- "This email and any attachments are confidential…"
- "Please consider the environment…"
- "You received this email because…"

Anchor matched, drop everything below.

What lands at the brain

By the end of stage 4, what's left is what the sender meant to write. A two-line question. A three-line request. A short paragraph asking about pricing. The brain reads *this*, plus the sender's name and email address from the envelope, and that's the input it makes a decision on.

A small concession: the reader stores the original raw email in S3, alongside the cleaned version. If a reply later turns out to be wrong because something important was in the signature or the quoted history, the audit trail has both. Cleanup is conservative, not destructive.

Attachments and weird shapes

The reader handles three attachment shapes pragmatically:

- **Images** — the file is stored in S3, the body keeps a placeholder "[image: filename]" so the brain knows there was one.
- **PDFs and documents** — same treatment; if your business actually needs the contents, that's a separate workflow (and the document pipeline series covers exactly this).
- **Calendar invites and vCards** — flagged in the envelope and routed to escalate by default. The brain doesn't accept invites on your behalf.

In plain words

Real email is messy. Most of what arrives is noise that humans skip without thinking, but a language model would patiently read every word of. The reader does the skipping first, in cheap plain code, and hands the brain only what the sender meant to say. Smaller input, sharper output, lower bill.

PART 4 OF 7

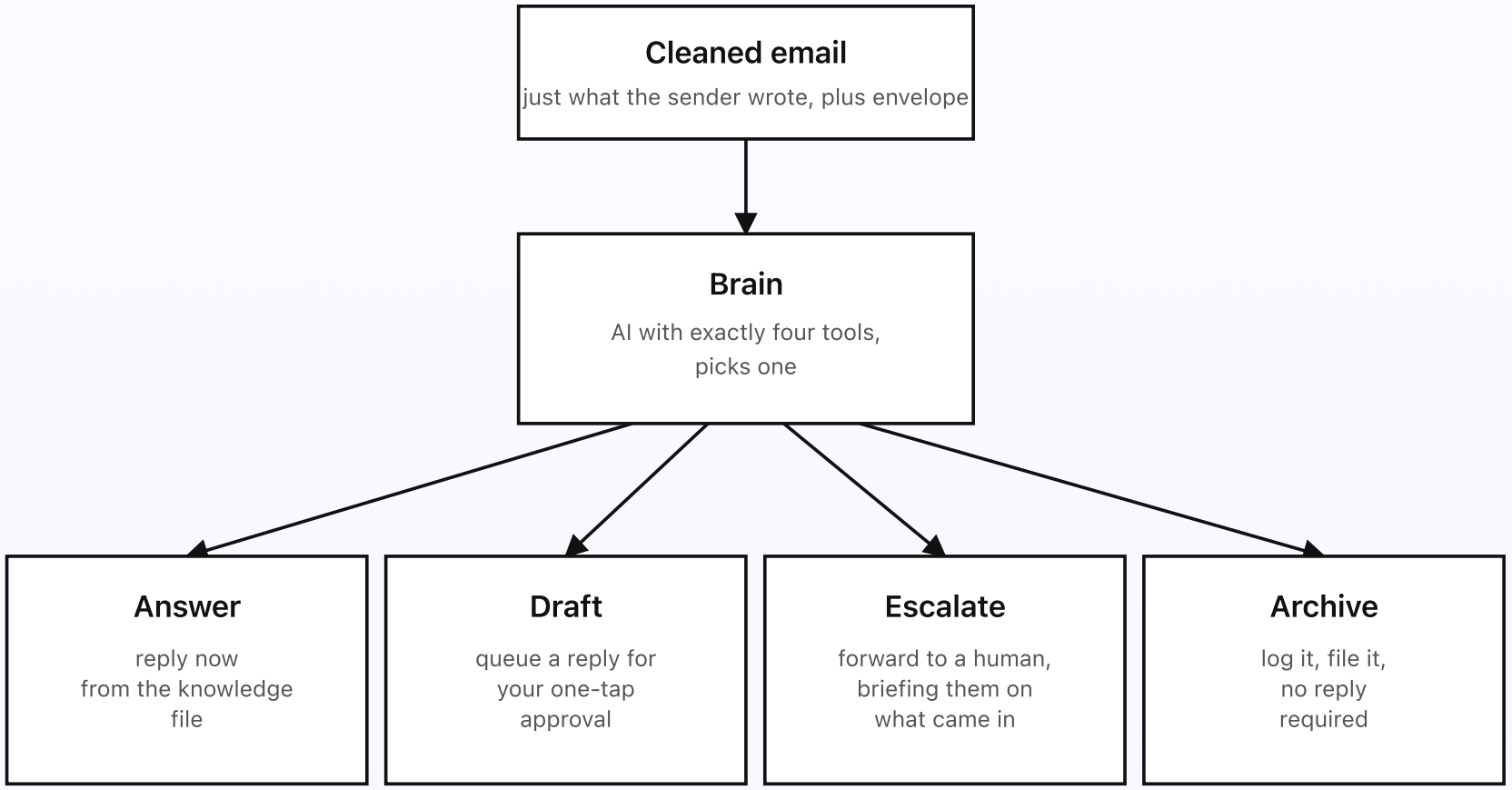
APRIL 29, 2026 PART 4 OF 7 · EMAIL ASSISTANT SERIES ~5 MIN READ

How the assistant decides what to do

The brain reads the cleaned email and picks one of four things: answer directly from the knowledge file, draft a reply for your review, escalate to a human, or archive without replying. It's allowed to be confident or to defer — never to invent.

KEY TAKEAWAYS

- The brain has a fixed menu of four tools per email: *answer directly*, *draft for review*, *escalate to a human*, *archive without replying*. There is no fifth.
- Strict tool_use makes free-text replies mathematically impossible — the model can only emit a structured tool call with a required `confidence_score` and `citation_passage_id`.
- *Answer directly* requires both a supporting passage in the knowledge file and a confidence score above the threshold; miss either and it locks to *draft for review*.
- Escalation forwards the original email plus a brief: what the sender wanted, the relevant knowledge passages, and the brain's best draft — the human walks into context, not a cold thread.
- Typical mature mix once the knowledge file is built up: 30–50% answer, 30–40% draft, 10–20% escalate, 5–15% archive — shifting toward answer as approved drafts feed the file.



The brain picks one tool per email. It can be confident or defer — never invent.

Fig 4. Four tools, one pick per email. The brain can't answer outside this menu.

Four tools, one decision

The most important rule for the brain is what it's allowed to do. Most "AI for email" tools fail because they're given too much freedom — they make up prices, promise discounts you'd never approve, agree to refunds without checking. Here, the brain has exactly four tools per email, and nothing else. If it can't fit the sender's message into one of these four shapes, it picks tool 3.

Tool 1 — Answer directly

The fast lane. The brain looks up the sender's question in the knowledge file (your hours, your services, your prices, your common FAQs), writes a short reply in your tone, and SES sends it. The reply threads correctly under the original message, and the sender sees a polite, accurate answer in their inbox within seconds.

The brain is only allowed to use this tool when two things are both true: the answer is supported by a passage in the knowledge file, and the model is confident in the match. If either is missing, it picks tool 2 instead. Confidence isn't a feeling here — it's a number the brain has to send back with its tool choice. Below the threshold, this tool is locked.

Tool 2 — Draft for review

The middle lane, where most real mail lands. The brain writes the reply, but instead of sending it, it puts the draft in a small queue you can clear from your phone in seconds.

The draft includes everything you need to make a one-tap decision: the cleaned email, the suggested reply, the knowledge passage the brain used, the confidence score, and any flagged concerns ("sender mentioned a refund — might want to check the policy doc"). You read it. Tap approve and the reply goes out. Tap edit, fix the bit that's off, send. Tap escalate and the email moves to your inbox.

This is the tool that turns the assistant from a risky auto-replier into a real assistant. Almost everything that isn't a textbook FAQ goes through here. You're still in the loop — you're just not typing.

Tool 3 — Escalate to a human

The escape hatch. The brain picks this when:

- The question isn't covered by the knowledge file at all.
- The sender is upset, asking for a refund, or asking about a sensitive topic (legal, medical, contractual).
- The sender explicitly asks to talk to a person.
- The email is a real-world request that needs action, not just an answer ("please change my appointment", "can you ship to this address instead").
- The brain isn't confident in any other tool.

Escalation forwards the original email to your team's inbox, with a short auto-generated brief at the top: what the sender wanted, what relevant knowledge

passages exist (so the human doesn't have to dig), and what the brain's best guess at a draft would have been. The human walks into a brief, not a cold thread.

Tool 4 — Archive (no reply)

For mail that doesn't want or need a reply at all. Examples that often land here:

- Cold outreach pitches from sales tools.
- "Just checking in" emails from vendors.
- Notifications that snuck through the noreply filter.
- One-line "thanks!" replies that close out a thread.

The brain logs the message and the reason in the audit table, and the lane ends. No reply, no draft, no notification. The audit log is searchable, so if you ever need to confirm whether the email arrived, you can — but it's not stealing your attention in the meantime.

What it never does

A short list of things the brain refuses to do, by design:

- Quote a price, hour, or policy that isn't in the knowledge file.
- Promise anything — a refund, a callback, a delivery date — without explicit support in the file.
- Accept calendar invites or commit to meetings on your behalf.
- Click any link in the email, including unsubscribe links.
- Continue a thread when it's clearly out of its depth.

If the sender pushes for any of these, the brain picks tool 2 or tool 3. The shape of the four tools makes this the only natural outcome — there's no "wing it" option in the menu.

How the four tools split your inbox in practice

A rough volume mix at small-business scale, after the rules in part 2 have already filtered out noreply and VIP traffic:

- **Tool 1 — Answer:** 30–50% of remaining mail at a mature business with a good knowledge file. The same five questions, asked five hundred ways.
- **Tool 2 — Draft:** 30–40%. In-between cases, longer requests, anything where confidence is good but not great.
- **Tool 3 — Escalate:** 10–20%. The work that really needs you.
- **Tool 4 — Archive:** 5–15%. The stuff that didn't deserve to reach a human at all.

The exact mix shifts as your knowledge file grows. The first weeks lean heavy on tools 2 and 3 (the brain hasn't seen enough yet); by week six, tool 1 is the largest group. Every approved draft is a small note to the future about what kind of question can be auto-answered.

In plain words

The brain has a small, fixed menu. Every email is a pick from four tools. The AI is fast and flexible inside that menu — it can phrase the same answer ten different ways, depending on the sender's tone — but it never tries to do something that's not on the menu. Boring, safe, correct.

PART 5 OF 7

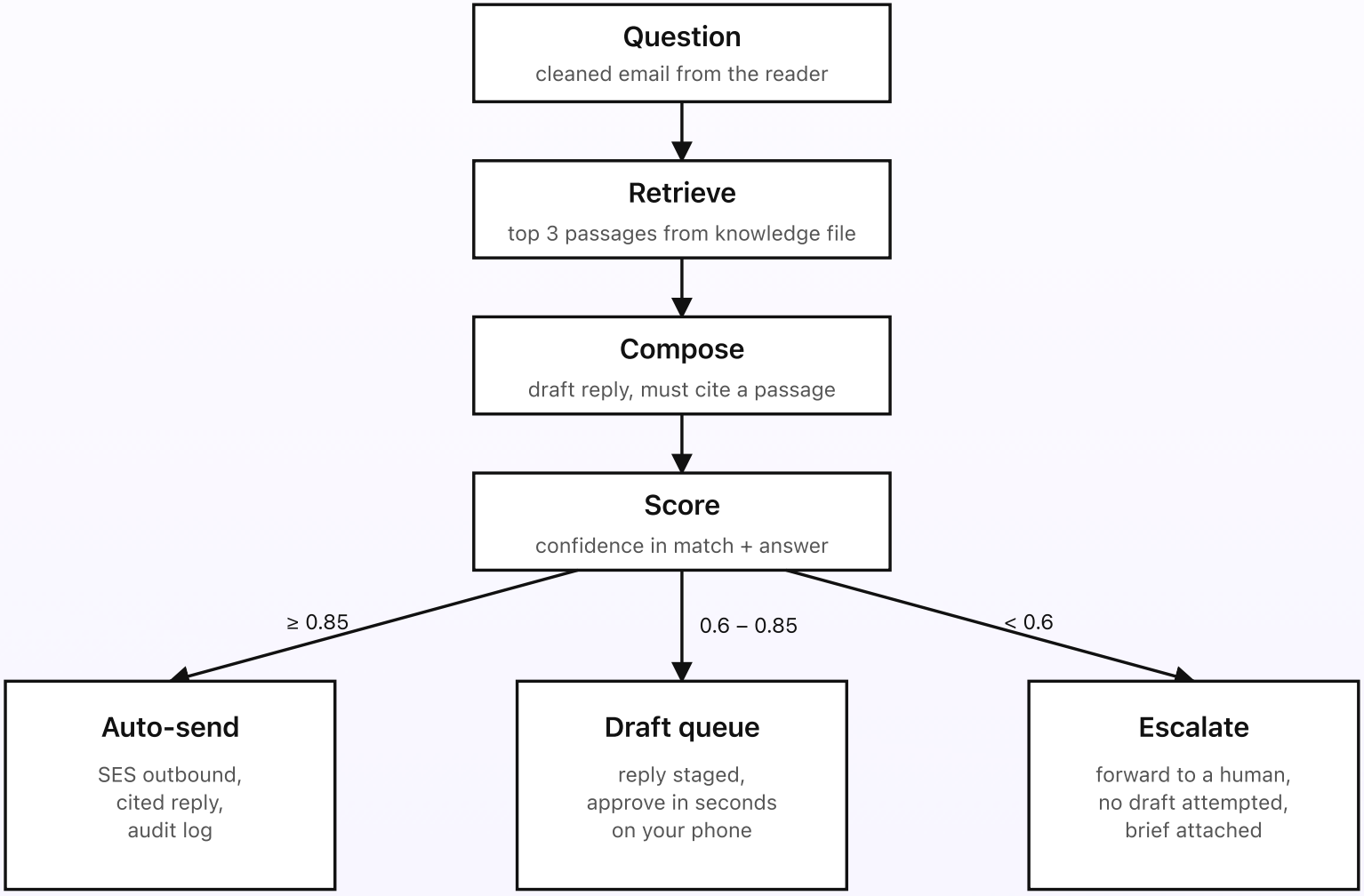
APRIL 29, 2026 PART 5 OF 7 · EMAIL ASSISTANT SERIES ~5 MIN READ

How a reply stays accurate

An auto-reply is only safe if it comes from something you wrote. The brain answers from passages in your knowledge file, points at the one it used, and refuses to send if it can't. Two thresholds, one safety net, no made-up prices.

KEY TAKEAWAYS

- Two gates before any auto-send: a passage in the knowledge file must support the reply, and the model's confidence score must clear the bar. Both, every time.
- The knowledge file is one Google Doc — about, services and pricing, FAQs, policies, tone — chunked, embedded with Bedrock Titan, and stored in **Amazon S3 Vectors** (GA December 2025, up to 2 billion vectors per index).
- Retrieval is three steps per email: search the top three passages, read with the strict instruction to answer only from those, and emit a `citation_passage_id` the runtime verifies.
- Two confidence thresholds split the score into three lanes: **0.85+** auto-send, **0.6–0.85** draft for human approval, **below 0.6** escalate. The thresholds live in the same Drive doc, no deploy.
- Self-correction is built in: replying from your own inbox marks the thread human-led so the assistant stops auto-sending, and editing the Drive doc fixes the next email of the same shape.



Every auto-reply has a citation. No citation, no auto-send.

Fig 5. Two thresholds. The brain has to be earning its auto-send privilege every single time.

The two questions every reply has to answer

Before any auto-reply leaves your domain, the brain must say yes to two questions:

1. **Is the answer in the knowledge file?** The brain has to point at a specific passage that supports its reply. If it can't, no draft gets written — the email goes to escalate.
2. **Is the model confident?** The model has to send back a number with its tool choice. Below the threshold, the reply becomes a draft for human approval, not an auto-send.

Both have to clear, every time. The model is fully allowed to say "I don't know" — it just turns into a different tool call.

The knowledge file is one Drive doc

Your "knowledge file" isn't a database, an admin panel, or a SaaS dashboard. It's one Google Doc you edit like a normal document, with sections roughly like:

- **About the business** — what you do, who you serve, where you're located, hours.
- **Services and pricing** — what you offer, what it costs, what's included.

- **Common questions** — the questions you've answered ten times each, with the answers in your tone.
- **Policies** — refunds, cancellations, lead times, what you do and don't support.
- **Tone and style** — how you want replies to sound. Formal? Warm? First-name basis? Always sign off with your name?

Behind the scenes, when you save the doc, a small sync job pulls the latest version, splits it into chunks, turns each chunk into numbers a computer can search (an embedding), and saves the result to a vector index in S3 (S3 Vectors — AWS's built-in vector storage, generally available since December 2025 at re:Invent, with up to 2 billion vectors per index). The brain searches that index for every email. There's no separate admin screen to keep in sync; the doc is the source of truth.

How retrieval actually works

For each cleaned email, the brain runs three short steps:

1. **Search.** Turn the question into numbers, look up the top three matching passages in the vector index. Cheap and fast.
2. **Read.** The brain reads the question and the three passages together. It's told plainly: only answer based on these passages; if they don't cover the question, say so.
3. **Cite.** If the brain writes an answer, it has to attach the ID of the passage it used. The audit log keeps both the reply and the citation, so you can check any auto-send by reading the source it came from.

The citation is what keeps things honest. A model with no requirement to cite will happily make up prices and policies. A model that must cite either points at a real passage or admits the question isn't covered.

The two thresholds

Confidence is a number, not a feeling. The brain sends back a score between 0 and 1 with its tool choice. Two thresholds split that range into three lanes:

- **0.85 and above — auto-send.** The brain found a clean match in the knowledge file and the answer is almost a copy of the cited passage. These tend to be FAQ-shaped questions: "what are your hours?", "do you ship to X?", "how do I reschedule?"
- **0.6 to 0.85 — draft.** The brain has a relevant passage but had to do real writing. Maybe the question covers two topics, or the passage is close but not exact. The reply is held for human approval, not sent.
- **Below 0.6 — escalate.** No good match. The brain doesn't even try a draft — the email goes straight to a human, with the brief from part 4 attached.

The thresholds can be changed and live in the same Drive doc as the knowledge file. Tighten them when you're launching, loosen them once you trust the assistant.

What happens when the model is wrong anyway

Even with citations and confidence thresholds, the model will sometimes auto-send a reply you wouldn't have. Three layers handle that:

- **Audit log review.** Every auto-send writes the question, the cited passage, the reply, and the score to a small table. Once a week, a short report lists the auto-sent replies so you can scan for misses.
- **Self-correction.** If you spot a bad reply, you reply to the original sender from your own inbox — the system sees that thread already has a human in it and stops auto-sending on it.
- **Knowledge-file edits.** A bad reply almost always traces back to a missing or unclear passage. Edit the doc, and the next email of that shape gets the better answer. The system improves itself by improving its source.

What it never does

The accuracy rules cash out as a few hard refusals the brain enforces every email:

- No reply without a passage citation. If the search comes back empty, the brain escalates.
- No made-up numbers. If the sender asks “how much” and the file doesn’t say, the reply asks them what package they have in mind, or escalates.
- No promises about future actions. The brain answers questions; it doesn’t book, schedule, or commit on your behalf.
- No claims of identity beyond what’s in the file. If the file says “Allan replies personally,” the assistant signs off as Allan; if not, it signs off in a general way.

In plain words

The reply has to come from something you wrote. The brain has to point at the passage. The score has to clear the bar. If all three line up, the reply auto-sends and you barely notice it happened. If any of them slip, the email is on your desk before it ever leaves your domain. The model gets to be helpful and confident inside the file, and silent outside it.

PART 6 OF 7

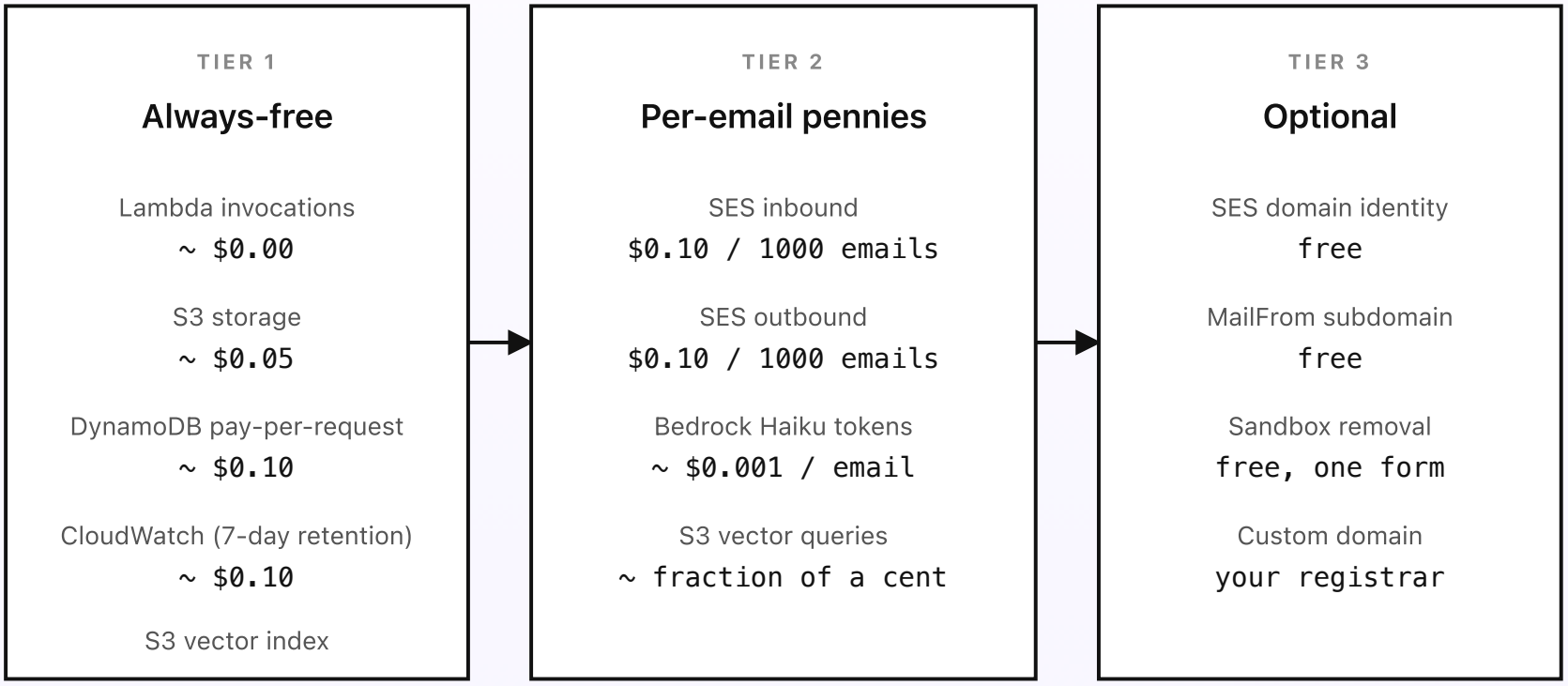
APRIL 29, 2026 PART 6 OF 7 · EMAIL ASSISTANT SERIES ~3 MIN READ

What the email assistant costs

A coffee a month at typical SMB volume. The fixed cost is roughly nothing. Everything else scales with the number of emails actually handled by the brain — not the noisy ones the routing rules already filtered out.

KEY TAKEAWAYS

- Fixed cost is roughly zero: Lambda, S3, DynamoDB, and CloudWatch all sit inside always-free quotas at SMB volume; quiet weeks bill basically nothing.
- SES inbound is about **$0.10 per 1,000 chunks** of 256 KB — a 200-emails-a-day inbox runs roughly 60¢/month.
- SES outbound is **$0.10 per 1,000 emails sent**, and only 30–50% of brain-handled mail auto-sends, so outbound stays in pennies.
- Bedrock Haiku tokens run **$1–$3/month** for a 200-emails-a-day inbox after the cleanup in Part 3 cuts inputs five-to-ten-fold.
- Most small businesses land between **$2 and $5/month**; a $10 AWS Budgets alarm catches anything strange, and 7-day CloudWatch retention from day one keeps log storage from drifting.



~ 200 emails a day → under five dollars a month, often under three.

Fig 6. Three tiers of cost. The bill scales smoothly with email volume; the floor is nearly zero.

The fixed cost is roughly nothing

Unlike the voice agent in the previous series, the email assistant has no fixed monthly floor. There's no phone number to rent. If the inbox is quiet for a week, the bill for that week is basically zero. Lambda, S3, DynamoDB, and CloudWatch all sit in the always-free tier at the volumes the assistant uses, and the vector index for the knowledge file is small enough to round to a few cents a month.

The flat costs you do pay are the AWS account's background ones — small storage fees, a handful of Lambda runs from the sync job, log retention. Together, well under a dollar a month at small-business volume.

The variable cost is per-email pennies

Three things scale with email volume:

- **SES inbound** — about $0.10 per 1,000 incoming chunks, where one chunk is 256 KB. A short text email is one chunk; a long thread or a heavy attachment can be a few. At 200 short emails a day, that's about 60 cents a month.
- **SES outbound** — about $0.10 per 1,000 emails sent. The auto-reply rate from part 4 is roughly 30–50% of brain-handled mail, so outbound is much smaller than inbound. Pennies a month at small-business volume.
- **Bedrock Haiku tokens** — roughly a fraction of a cent per email after the cleaning and search steps from posts 3 and 5 cut the input down. A 200-

emails-a-day inbox usually lands at one to three dollars of Bedrock spend per month.

Add it up: most small businesses end up between $2 and $5 a month total. The assistant pays for itself the first morning you don't spend an hour answering "what are your hours?" for the tenth time.

Three traps you're avoiding

- **Per-seat AI email tools.** Most "AI email assistant" products charge $20–$50 per inbox per month, no matter how much you use them. You're trading a flat subscription for pay-per-use that mostly comes in pennies.
- **Sending the whole thread to the model.** The reader strips quoted history before the brain sees anything. A simple setup that doesn't do this pays the model to read every previous reply quoted back at it; you don't.
- **Logs that quietly grow.** CloudWatch keeps logs for only 7 days from day one. Without that, log storage slowly drifts up forever. Seven days is enough to debug live, and the audit table carries the long-term record.

When this stops being cheap

The math changes at high volume. A busy support inbox at 5,000 emails a day with a draft-most policy might land at $30–$60 a month — still cheaper than the per-seat products, but no longer coffee money. At that point the cost is mostly model tokens, and switching the brain to a smaller model or tightening the search pays off quickly.

For everyone below that — and that's most small businesses — the bill is small enough that a $10 monthly AWS Budget alarm catches anything strange before you'd notice on the credit card.

In plain words

The fixed bill is nearly zero. The variable bill is cents per email. A typical small-business inbox runs at coffee-money for the whole month. Set a budget alarm that fits your expected volume and the bill can't surprise you.

PART 7 OF 7

APRIL 29, 2026 PART 7 OF 7 · EMAIL ASSISTANT SERIES ~3 MIN READ

Engineering reference: the email assistant architecture

Same system as the rest of the series, drawn purely for engineers. Service names, resource identifiers, region, Bedrock model IDs, and the actual flow operations — everything you'd need to recreate this in your own AWS account.

KEY TAKEAWAYS · VERIFIED MAY 2026

- Single AWS account in `ap-southeast-1` (Singapore); Bedrock via Global cross-Region inference.
- Five subsystems: Build & Deploy, Config Sync, Reader (SES inbound → S3 → `fn-reader`), Brain (`fn-brain` with strict tool_use), Sender (`fn-sender` routes by tool choice).
- Models: `global.anthropic.claude-haiku-4-5-20251001-v1:0` + `amazon.titan-embed-text-v2:0`; vector store is the `vec-knowledge` S3 Vectors index (GA Dec 2, 2025).
- DynamoDB: `tbl-messages` (envelope + decision), `tbl-drafts` (pending approvals), `tbl-audit` (every action). Cross-cutting: 7-day CloudWatch retention, $10 AWS Budget, weekly archive to S3 Glacier IR.
- Email deliverability paperwork on day one: SES domain identity with DKIM, custom MailFrom subdomain for SPF/DMARC alignment, and SES sandbox removal request.

Posts 1–6 walk through the system in plain language. This page is the dense version — nothing softened, just the architecture as you'd sketch it on a whiteboard during a design review.

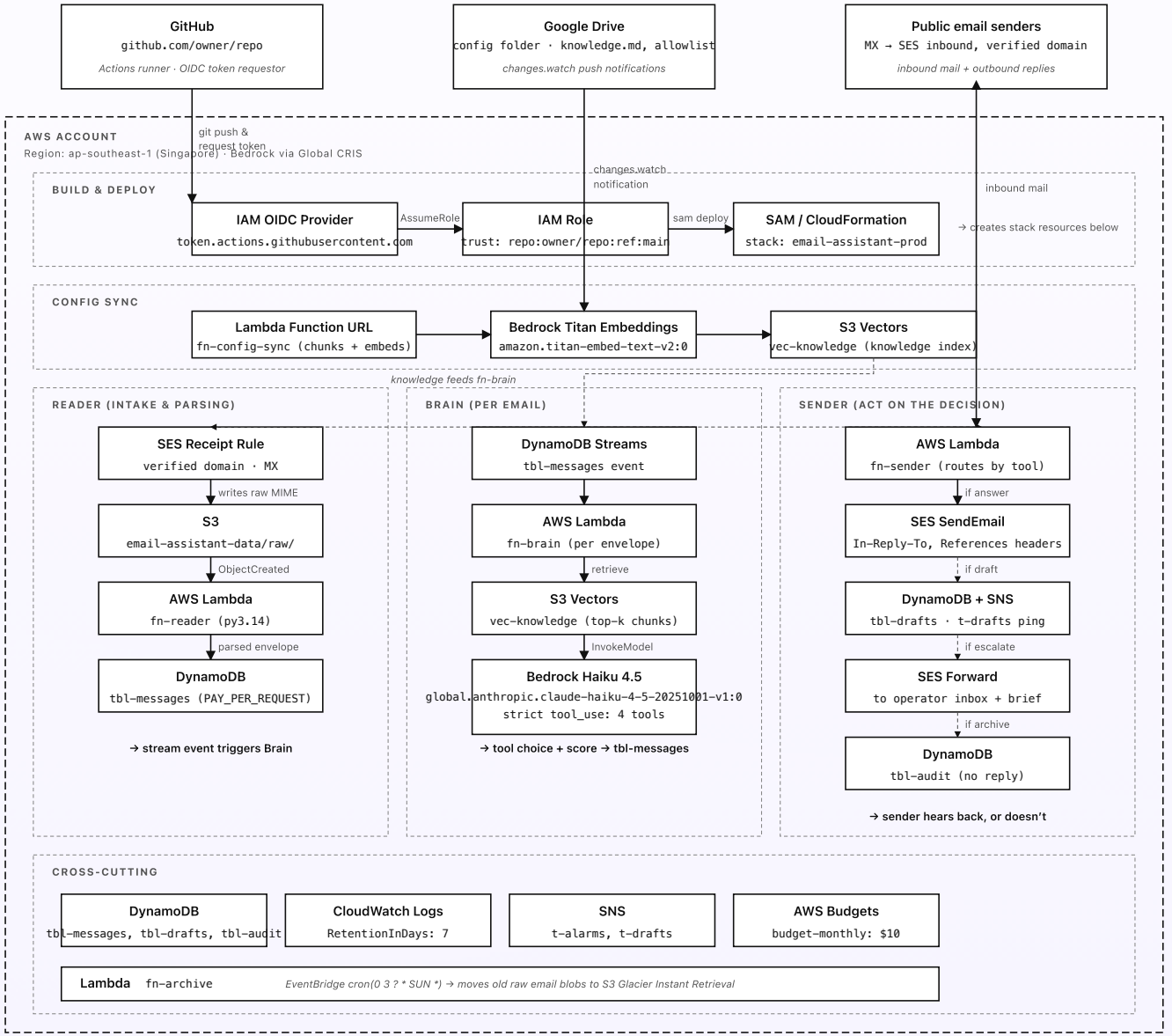

GitHub
github.com/owner/repo
Actions runner · OIDC token requestor
Google Drive
config folder · knowledge.md, allowlist
changes.watch push notifications
Public email senders
MX → SES inbound, verified domain
inbound mail + outbound replies
AWS ACCOUNT
Region: ap-southeast-1 (Singapore) · Bedrock via Global CRIS
git push & request token
changes.watch notification
inbound mail
BUILD & DEPLOY
IAM OIDC Provider
token.actions.githubusercontent.com
AssumeRole
IAM Role
trust: repo:owner/repo:ref:main
sam deploy
SAM / CloudFormation
stack: email-assistant-prod
→ creates stack resources below
CONFIG SYNC
Lambda Function URL
fn-config-sync (chunks + embeds)
Bedrock Titan Embeddings
amazon.titan-embed-text-v2:0
S3 Vectors
vec-knowledge (knowledge index)
knowledge feeds fn-brain
READER (INTAKE & PARSING)
SES Receipt Rule
verified domain · MX
writes raw MIME
S3
email-assistant-data/raw/
ObjectCreated
AWS Lambda
fn-reader (py3.14)
parsed envelope
DynamoDB
tbl-messages (PAY_PER_REQUEST)
→ stream event triggers Brain
BRAIN (PER EMAIL)
DynamoDB Streams
tbl-messages event
AWS Lambda
fn-brain (per envelope)
retrieve
S3 Vectors
vec-knowledge (top-k chunks)
InvokeModel
Bedrock Haiku 4.5
global.anthropic.claude-haiku-4-5-20251001-v1:0
strict tool_use: 4 tools
→ tool choice + score → tbl-messages
SENDER (ACT ON THE DECISION)
AWS Lambda
fn-sender (routes by tool)
if answer
SES SendEmail
In-Reply-To, References headers
if draft
DynamoDB + SNS
tbl-drafts · t-drafts ping
if escalate
SES Forward
to operator inbox + brief
if archive
DynamoDB
tbl-audit (no reply)
→ sender hears back, or doesn't
CROSS-CUTTING
DynamoDB
tbl-messages, tbl-drafts, tbl-audit
CloudWatch Logs
RetentionInDays: 7
SNS
t-alarms, t-drafts
AWS Budgets
budget-monthly: $10
Lambda fn-archive
EventBridge cron(0 3 ? * SUN *) → moves old raw email blobs to S3 Glacier Instant Retrieval

Fig 7. Full architecture, ap-southeast-1. White boxes = AWS resources; dashed AWS container; dashed grey boxes = subsystem groupings; dashed grey arrows = config feed and side branches.

Read this top-down, then column-by-column

Top row is the three external surfaces. Below it, the AWS account contains five subsystems: Build & Deploy across the top, then Config Sync, then three runtime columns (Reader, Brain, Sender), with a Cross-cutting strip at the bottom. Inbound mail arrives at SES on the verified domain, lands in S3, fans out to `fn-reader`, which writes a parsed envelope to `tbl-messages`. The DynamoDB stream then triggers `fn-brain`, which retrieves passages from S3 Vectors and invokes Bedrock with strict tool_use over four tools. The choice and confidence score get written back to `tbl-messages`, which triggers `fn-sender` — routing to SES outbound, the draft queue, escalate-forward, or just an audit row, depending on which tool fired.

Naming conventions used in the diagram

- **Lambda functions:** `fn-<purpose>` — `fn-reader`, `fn-brain`, `fn-sender`, `fn-config-sync`, `fn-archive`.
- **DynamoDB tables:** `tbl-messages` (every inbound, with parsed envelope and current state), `tbl-drafts` (pending human approvals), `tbl-audit` (every action ever taken).
- **SNS topics:** `t-alarms` for general failures, `t-drafts` for pending-draft pushes to the operator's phone.

- **S3 layout:** single bucket `email-assistant-data` with prefixes `raw/{date}/`, `config/`, `archive/`.
- **S3 Vectors index:** `vec-knowledge` — chunked + embedded knowledge file for top-k retrieval.

Region, model access, and email deliverability

Everything runs in `ap-southeast-1` (Singapore) for low latency from the Philippines. Bedrock model invocations use the **Global cross-Region inference** profile (model IDs prefixed with `global.`) — data at rest stays in Singapore; inference may route to other regions for capacity. Pricing is the same as on-demand Singapore pricing.

Email deliverability is the part of email that catches out engineers who are used to webhooks. Three pieces have to be set up before any reply leaves your domain:

- **SES domain identity** — verify the domain via DKIM CNAMEs in your DNS provider. Verified domains can both receive (with the right MX records) and send.
- **Custom MailFrom subdomain** — `mail.yourdomain.com`, with its own SPF/MX records, so SPF and DMARC alignment work correctly when SES sends on your behalf.
- **SES sandbox removal** — new SES accounts are sandboxed (can only send to verified addresses). One short request form on the AWS console moves you to production limits.

The brain uses **strict tool_use**: four tool definitions (`answer_directly`, `draft_for_review`, `escalate_to_human`, `archive_no_reply`) with required parameter schemas including a `confidence_score` in `[0, 1]` and a `citation_passage_id` string. So the model can only emit a structured tool call — not a free-text reply. Free text would let it invent prices or promises; tool_use makes that mathematically impossible.

What's deliberately not on the diagram

- IAM policy details — per-Lambda execution role inline policies are minimal (one bucket prefix, one or two tables, SES on a single sending identity, Bedrock invoke on one model).
- Per-business knowledge schema — the `knowledge.md` file is a single Drive doc with sections for FAQs, hours, services, policies, and tone. Updating sections updates the assistant's answers without a deploy.
- X-Ray tracing — on for `fn-brain` and `fn-sender`, sampling 100% during tuning, 10% in steady state. Mostly for confidence-score calibration, not latency.
- The CloudFormation parameters for the Bedrock model ID and the embedding model are templated, so swapping models doesn't require code changes.
- **Amazon WorkMail** — AWS's managed inbox product. If you want the operator's "normal inbox" to also live in AWS, this slots in next to SES; for most teams, the operator inbox is just the existing Gmail or Outlook setup and no WorkMail is needed.

- **Bedrock Knowledge Bases** — managed retrieval that can replace the explicit S3 Vectors + Titan Embeddings setup with a single connector. Bedrock Knowledge Bases now natively supports S3 Vectors as the storage engine, so the underlying primitives are the same; you just get an opinionated chunk-and-retrieve loop instead of running it yourself. Worth picking when the bring-your-own-knowledge-file pattern isn't strictly required.
- **Bedrock Guardrails contextual grounding check** — Bedrock's managed grounding-and-relevance scoring layer. The custom `confidence_score` and `citation_passage_id` tool parameters in `fn-brain` are roughly the same idea hand-rolled; swapping to Guardrails moves the thresholds into console configuration and adds a second pair of eyes (PII redaction, prompt-attack detection, topic filters) on every model call. Worth turning on once the in-code thresholds are stable.
- **SES Mail Manager** — the managed inbound rule engine. It can host the lane-classification logic that `fn-reader` does today, with rule actions including invoking a Lambda for the AI-handle case, and is a clean drop-in once the receipt-rule library grows beyond a handful of patterns. The traditional SES Receipt Rules path used in the diagram remains valid; Mail Manager is the upgrade, not a deprecation.

IF YOU'RE RECREATING THIS

Start with Build & Deploy alone (a single Lambda, no triggers). Once `git push` reliably updates an empty stack, verify a domain in SES and get a single "hello" auto-reply going on a hard-coded subject line. Then S3 inbound storage. Then `fn-reader` with just lane classification (no parsing yet). Then the parsed envelope. Then the Brain on a single hard-coded tool. Then the other three tools. Then the draft queue and the operator notifications. Cross-cutting (audit, logs, alarms, budget, archive) goes in from day one.