26 Grok Bots worth copying

Sorted by job. What each one does, how it runs underneath, where it breaks, and a check before you trust it.

So this comes from Divyanshi Sharma's list of 26 Grok Bot templates. It's a really good list. I opened every bot's public page, read what its maker wrote, and took each one apart.

A Grok Bot is basically a saved workflow with one job. One input goes in, one useful output comes out.

I sold a Grok Bot setup for $2.5k. So this is how I'd look at all 26.

How to read each entry:

- **What it does:** the job, taken from the bot's own public page and put in my words. The bot's maker is named next to it.
- **How it runs:** my read of the steps underneath. Two bots (Echo and Copay Compass) publish their full instructions on the page, so theirs are close to exact. For the rest, the page gives the job and I filled in the steps.
- **Where it breaks** and **Check:** mine. Run the check before you trust what it hands back.

What the best ones have in common

Reading all 26 pages side by side, the strongest bots share four habits.

1. **They write down what they never do.** Copay Compass never submits anything and never asks for an ID number. Video Editor never posts and never overwrites your files. Echo never makes up what was said on a call. Skippy never says a parking spot is legal.
2. **They set up with questions, one at a time.** Echo asks for your template, your notes tool and your rules, one question each, then saves the answers. Dial Bot asks for its phone API key and a voice on first use. Nightly Audit Engineer asks when to run.
3. **A person does the last step.** Engineer Bot leaves the merge to you. Tech Demos asks before it builds. Copay Compass prepares the application and the patient sends it. Video Editor stops at a master for review.
4. **They cap their own scope.** The Page watches 3 to 5 pages. Be Happier gives 3 suggestions a week. Tech Demos builds one library a day.

1. Watch and alert (3 bots)

The Page (by Hiten)

What it does: You give it 3 to 5 public pages and the one detail you care about on each. It checks every morning, and you only hear from it if that detail moved. Day one you get a short list of what it's tracking.

How it runs: you give it the pages and the one thing that matters on each. Every morning it fetches each page, compares that part with yesterday's copy, and decides if it really moved.

Where it breaks: silence looks the same whether nothing changed or the bot stopped. Banners, dates and layout tweaks can look like a change.

Check: watch a page you control, change the thing it watches, and make sure a message lands the next morning. Then change something irrelevant and make sure nothing comes.

When It Matters (by Hiten)

What it does: You name something you keep refreshing, like a launch date or a price. It figures out which signal would settle the question, tracks that, and pings you once it does. If you have nothing in mind, it can suggest open threads from what you already follow.

How it runs: turn your question into one deciding signal, check that signal on a schedule, and ping you once it crosses the line you agreed on.

Where it breaks: it can pick the wrong signal and watch it very confidently. "When it matters" is its own judgement, so it can be too chatty or too quiet.

Check: ask it to name the exact signal and the line that triggers a message before it starts. Test it on something you know is about to change.

Talent Matchmaker (by Lenny)

What it does: Goes through your inbox for two things: roles companies are hiring for (often buried in investor updates) and people who've written in looking for a job. It keeps both lists current and pairs them up.

How it runs: scan the inbox, pull out open roles and job seekers, keep both lists current, then pair them up.

Where it breaks: it reads your whole inbox, and investor updates are often private. Roles go stale. A casual line can get read as "looking for work".

Check: run it on a month of email you know well. Ask for the source email behind every match, and confirm the role is still open before any intro.

2. Read something, give a verdict (6 bots)

Pattern of Pain (by Hiten)

What it does: Hand it one company's website. It digs out the complaints customers raise again and again, backs each one with proof, and points you at what to dig into next.

How it runs: work out what the company sells, gather what customers say about it, group the complaints, keep only the ones that repeat, and attach the evidence to each.

Where it breaks: sites with little public customer talk give thin patterns. One loud review can pass as a pattern.

Check: open every evidence link and confirm the quote is really on that page. Ask for 2 or more separate sources per pain. Run it on a business you know well first.

X Algo (by Matt)

What it does: For people who post on X: should you quote this, post something fresh, or hold off. It bases the call on the open-source code behind X's For You feed and the post's current numbers.

How it runs: read the live numbers on a post, compare them with what the ranking code rewards, and decide if the post is still climbing or has peaked.

Where it breaks: the public code may not match what X runs today. The call can sound certain when it's really a guess, and small accounts don't have enough data.

Check: log every call it makes, act on it, and compare with what happened 24 hours later. After about 20 calls, see if it beats your own gut.

Pitch Deck Coach (by Hiten)

What it does: Reads your deck the way an investor would. It tells you which points land, which ones people will doubt or push back on, and what sticks after the meeting, then helps you fix the narrative, the proof and the slides.

How it runs: read the deck slide by slide through those four lenses, flag weak claims and missing proof, then suggest fixes.

Where it breaks: advice that fits any deck. It can also push for evidence the founder doesn't have.

Check: run it on 2 or 3 decks where you know the real investor reaction. Every suggested piece of evidence must be something the founder actually has.

Product Idea Stress Test (by Hiten)

What it does: Paste in an idea for a product. It spells out the conditions the idea depends on, the evidence on both sides, the single assumption that would sink it, and the lowest-cost experiment to check it.

How it runs: break the idea into assumptions, look for evidence on each, rank them by what sinks the idea fastest, and design one small test.

Where it breaks: it can pick a safe assumption instead of the real killer. Evidence can be thin if it isn't actually searching.

Check: every piece of evidence needs a source you can open. The test needs a pass or fail line you could run this week. Try it on an idea you already know failed.

Lennybot (by Lenny)

What it does: Searches the archive of Lenny's newsletter to answer your question. The first time you chat, it gets you to link your Lenny's account with an email login.

How it runs: connect, search the archive, answer from what it finds.

Where it breaks: if the search finds nothing, it may answer from general knowledge instead. It only knows one archive.

Check: ask something you know the archive covers and make sure it names the post. Ask something it can't cover and see if it says so.

Skippy (by Matt)

What it does: Built for San Francisco drivers. Drop in a location (a map pin, a street address or a corner) and it reads the city's public schedule to tell you when that curb is next swept. It will not tell you a spot is safe to park in.

How it runs: turn the location into a street, block and side, look up that curb in the city's sweep schedule, and work out the next date.

Where it breaks: a pin on the wrong side of the street. Holidays and temporary signs aren't in the schedule. One city only.

Check: test 10 curbs where you can read the sign, including both sides of the same street.

3. Turn one input into a finished thing (4 bots)

Echo (by Krista)

What it does: Takes the notes from a customer call (Granola or Gong) and fills in slides in Figma or Google Slides: a recap of the call, the agreed next steps, the projects that came out of it, or every one of those. Its page publishes its full setup.

How it runs: setup is a short interview, question by question. It studies some emails you've sent to pick up your tone, then asks for the slide to use as a layout, what belongs on it, any must-say or never-say rules, which notes tool you use, and your brand guide. It also asks whether to refresh the deck after each outside call. After a call, it holds off until the notes arrive, edits only the slides you picked, keeps your layout and tone, and opens with the client's problem. It won't invent quotes, won't add slides, and won't share the file.

Where it breaks: the calendar can mark a call as external when it isn't. Rewriting the same slides each call can overwrite last week's version. Wrong names in the transcript end up on the slide.

Check: run it on one past call you know well and check every line against the notes. Use the file's version history to confirm only the named slides changed.

Copay Compass (by Marc)

What it does: Helps a cancer patient cover the cost of their drug. It finds assistance programmes, sets up alerts for when a closed fund starts taking applications again, and drafts the forms and appeals. The patient sends everything. Its page publishes its full rules.

How it runs: first run asks for 5 things: the drug, the cancer type in general terms, the state, the insurance type and, if they want, the treatment center. Then it works out which kind of help fits the insurance, sets up fund alerts through tools that already exist, shows each programme's published rules with the link and the date checked, drafts the paperwork, and books a one-week check-in. It never says whether someone qualifies, never gives medical advice, never touches ID or bank numbers, and never sends anything itself. Anything sensitive goes on a handoff sheet for the hospital's financial navigator.

Where it breaks: programme rules change fast, so a fact checked last week can be wrong today. It's built for the US only.

Check: every programme fact needs a link and a date. Open two of them yourself. Confirm it never says "you qualify".

Connections (by Matt)

What it does: Takes everyone you follow on X, flags who follows you back, and sorts them by city into a Notion database that sits in your own workspace, so you can search it.

How it runs: pull your following list, mark who follows back, clean up each profile's location into a city, and write one row per person to Notion.

Where it breaks: lots of profiles have no location or a joke one. "SF" and "San Francisco" can end up as two cities. Reruns can create duplicates.

Check: pick 20 people whose city you know and see if they land right. Run it twice and make sure the row count doesn't grow.

Tinkabot (by lauren)

What it does: Packages an API as a Cursor plugin, a connector plus skills. Its order: understand the data, build the smallest version that runs, test it on your machine, check once who owns the API, then list it publicly.

How it runs: read the API's data, build the smallest working connector, test it on your machine, confirm ownership, publish.

Where it breaks: the smallest version can skip login refresh, paging and rate limits. Keys can slip into the published package.

Check: before publishing, call every tool with a test key, including one bad login and one paged call, and search the package for secrets.

---

4. Make creative work (5 bots)

Clip Bot (by Lon)

What it does: Feed it a YouTube interview or podcast and you get 16:9 highlight clips with word-by-word captions and a card crediting the source, saved straight to your folder.

How it runs: fetch the video, transcribe it with word timings, pick the moments, cut to 16:9, burn in the captions, add the source card, save.

Where it breaks: clips that start or end mid-sentence. Names spelled wrong in captions. Cutting someone else's interview may need their permission.

Check: watch every clip end to end. Full sentences, captions match the words, right source on the card.

Video Editor (listed as Best Video Editor, by X)

What it does: Edits footage you own or official footage: cutting, recutting, notes on shots, general help. It settles on the approach before touching anything, then delivers a master for you to review. It won't publish anything or touch your original files.

How it runs: confirm the footage is yours, decide the treatment, cut, export a new file, stop.

Where it breaks: if the plan is wrong, the whole cut follows it. Audio pops at the cut points.

Check: read its plan before it cuts. Then watch the master, listen at every cut, and confirm your original files haven't changed.

Writing Bot (by Matt)

What it does: Helps you draft or rework essays, emails and docs. It follows a fixed editing process to make the piece clearer and tighter while keeping what you meant, your facts and how you sound.

How it runs: read the draft, fix the structure, then clarity, then the lines.

Where it breaks: short drafts have little voice to keep, so it drifts into generic polish. Restructuring can quietly drop a point.

Check: put both versions side by side and list every fact in each. Nothing added, nothing dropped. Read it out loud.

Image Gen Bot (by Matt)

What it does: Makes images and short clips. It collects your brand material and reference images, writes the prompt for Grok Imagine, generates, removes the background using FAL, and repeats until you sign off.

How it runs: exactly those steps, with you approving each round.

Where it breaks: off-brand results even with references. Rough cut-out edges on hair and glass. Long loops add up in generation cost.

Check: a brand checklist (colours, logo, fonts, any claims), a zoom on the edges, and a cap on rounds before you start.

Figma Bro (by John)

What it does: Builds designs directly in Figma, with proper reusable components, a clean layout and animation.

How it runs: read the brief, plan the layout, build real reusable components, lay them out, add the transitions.

Where it breaks: without your design system it makes up its own styles. "Feels right" has no measure.

Check: confirm the parts are real components, not grouped shapes. Click through the prototype. A designer reviews the first one of each type.

---

5. Build and run code (6 bots)

Dr Eggbot (by lauren)

What it does: Builds new Grok Bots for you. It starts with a short set of questions about what you want, then makes the bot. For coding bots the bar is one job, tidy output and proof it works. Other bots get one job, one tone, a written list of jobs they refuse, and no tools they don't use.

How it runs: questions, pin the one job, write the "doesn't do" list, strip unneeded tools, create.

Where it breaks: vague answers give a fuzzy bot, and it acts as soon as it thinks the job is clear. "Verified" isn't defined.

Check: read the one-job line and the "doesn't do" list. Run 5 to 10 real inputs, including 2 it should refuse.

Loops (by Matt)

What it does: Sits on top of your coding agents and runs the loop for them. It writes each goal with a test that proves it's done, then collects context, prompts, launches, reviews and merges. It only works in a repo you name and won't pick one itself.

How it runs: gather context from the repo, write the goal with its proof, hand it to a coding agent, review against the proof, merge.

Where it breaks: a goal with no real test gives fake proof. No loop limit is stated, so a failing goal can keep spending.

Check: read the proof before launch and make sure it can actually fail. Keep merging behind your approval for the first few runs.

Engineer Bot (by Lingxi)

What it does: Supervises engineering work so you don't have to. It turns requests into board items, starts cloud agents on them, checks their pull requests twice an hour, and the only thing it needs from you is the merge.

How it runs: board the request, launch an agent per item, poll the pull requests, nudge on failures, ask you to merge when ready.

Where it breaks: "ready to merge" is its judgement. Agents running in parallel can edit the same files. No spend cap is stated.

Check: tests pass, a person reads the diff, and each pull request links to one board item with a clear done line.

Nightly Audit Engineer (by Lingxi)

What it does: Goes through your whole codebase overnight, then makes one cleanup in each area. It runs at 4am unless you pick another time, and asks you up front.

How it runs: read the codebase, split it into areas, pick one cleanup each, ship them.

Where it breaks: a "cleanup" that changes behaviour. Lots of small changes every night can bury the reviewers.

Check: keep it on a branch for the first week, require passing tests on every cleanup, and read every diff.

Tech Demos (by Matt)

What it does: Each weekday it looks through your X bookmarks, chooses one new library, checks with you, then plans and builds a demo of it inside a single shared repo, using a Cursor agent in the cloud.

How it runs: scan bookmarks, pick one, wait for approval, plan, build into the same repo every day.

Where it breaks: new libraries have thin docs, so the demo can target an old version. The shared repo gets messy over time.

Check: clone the repo fresh and run the new demo with one command.

Grok Build (by Teslaconomics)

What it does: Hands your task to the actual Grok Build command-line tool, at its highest effort setting, on a computer in the cloud. Use it for building apps, writing code or long research jobs, where you want real work done rather than a chat answer.

How it runs: take the task, run the CLI, hand back what it made.

Where it breaks: vague task in, vague build out. Maximum effort means slow, heavy runs, and the page states no limits.

Check: start with a small, clear task. Tests and a diff read for code, a source for every claim in research.

6. Act for you (2 bots)

Dial Bot (by Matt)

What it does: Calls people for you with an AI voice, using Bland, and tells you how each call went. The first time, it asks for your Bland key and which voice to use.

How it runs: turn your request into a call brief, place the call through Bland, pull the result back, report.

Where it breaks: a vague brief means the AI says the wrong thing to a real person. Calling rules differ by country and state. Every minute is billed to your key.

Check: make 3 test calls to your own phone, listen, and compare with what it reports. Set a spend limit on the key first.

Be Happier (by Lenny)

What it does: Once a week it goes through your inbox and calendar and suggests 3 specific changes to make your week better. The rule is to guard what you already have, not pile on new routines.

How it runs: read the week, spot what eats your time or keeps getting moved, pick 3 changes, drop anything that adds a habit.

Where it breaks: it reads private mail. It can misjudge a meeting from the calendar entry alone.

Check: each suggestion should name the exact meeting or thread behind it. Decide which folders it may not read before you connect it.

If I were starting, I'd copy these five

1. **The Page.** It runs while nobody is looking. Silence is the product.
2. **Pattern of Pain.** The evidence sits next to every claim, so it's the one I'd trust most.
3. **Echo.** A call already holds the facts people forget to write up, and its setup is the best one on the list.
4. **Nightly Audit Engineer**, pointed at a client's workflows and tracking instead of code.
5. **Clip Bot.** One long recording becomes a stack of short clips.

My setup notes

The same steps work for any of the 26.

1. **Pick one job.** Write it in one sentence: this goes in, this comes out. If the sentence needs an "and", it's two bots.
2. **Write down what it never does.** Copay Compass and Echo both do this on their page. Any bot that touches a customer needs a list like that.
3. **Fix the output shape.** Same fields on every run, for example: pain, quote, source, next step. Free text is hard to check.

4. **Write the check before the prompt.** Collect 5 to 10 sample inputs where you already know the right answer. Run them after every change.
5. **Connect only what it needs.** Start read-only. Add each account one at a time.
6. **Put a person before the customer.** Anything a client sees gets approved first. Drafts land in a folder. Nothing sends itself.
7. **Log every run.** Input, output, check result. That's how you know run 50 is as good as run 1.
8. **Watch the watcher.** If it runs on a schedule, have it send you a weekly "I ran" message. A monitor that dies quietly looks exactly like one with nothing to report.

One I built

So for example, for a marketing agency I built a workflow that takes a brand. It searches Amazon for the product, pulls 20 listings, scores them on relevance, stars and review count, and keeps the 2 closest competitors. Then it pulls their live Meta ads, 117 of them, and hands back a competitor report. The data cost $0.18, and it shows me the cost and sets a hard spending cap before any paid pull starts.

Same shape as Pattern of Pain. Pull the data, find what repeats, hand back a report. No rocket science.

A template gives you the steps. You bring the data and the checks. That's the whole point.

If you want one of these set up on your own data, send me a DM. I'll tell you which one I'd start with.

Abhinav Mukala. LinkedIn