CentralHost

FLEET MANAGEMENT GUIDE

One Console for Your Whole Fleet

Stop running your servers from a spreadsheet.

A field guide for agencies, MSPs, and developers running a fleet of Linux servers — without a hosting panel.

www.centralhost.sh

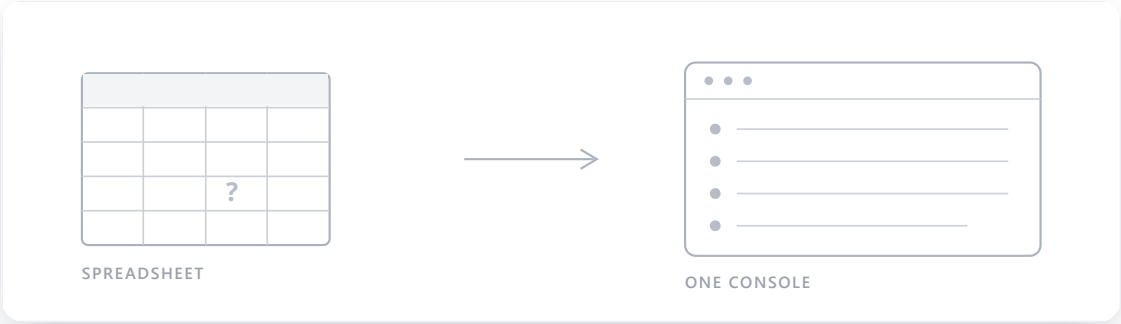
2026 Edition

Contents

Introduction · The fleet you can finally see

You're running more servers than you can hold in your head, and the only thing that knows the whole picture is a spreadsheet you've quietly stopped trusting. The machines are scattered across two or three providers, each with its own console that knows only its own boxes. Monitoring is a separate tool. Passwords are in a third. The thing tying all of it together — the part that knows the spreadsheet row, the alert, and the SSH alias all refer to the same server — is you, holding it in your memory and hoping nothing slips while you're not looking.

CentralHost is the thing that comes after the spreadsheet.



It's a single console for an entire fleet of Linux servers. You install one small agent on each machine and from that moment the server describes itself to you — what it is, what it's running, how it's doing, what's exposed — and keeps that picture current on its own. Every server you own, no matter which provider it lives on, shows up in one view that reflects what's true right now, not a snapshot that went stale the moment it was saved. Monitoring, access, security posture, patching: the five disconnected tools collapse into one place that actually agrees with reality.

This short book is the tour. It walks through what CentralHost does, in the order you'd actually feel it: first you can *see* the fleet, then you can *know* when something's wrong, then you can *take care of it* as a group instead of one box at a time, and finally you can *act* — open a terminal, or hand the investigation to an AI assistant that reads across every machine for you and comes back with the answer.

A word on that last part, because it's the part we're proudest of and the reason CentralHost is more than a nicer dashboard. The hardest thing about running a fleet was never knowing how to fix a server — you already know that. It was the *walk*: when something breaks on one of forty machines, finding which machine, which process, which line in which log is a manual slog that gets longer with every server you add. CentralHost's assistant does that walk for you. Point it at a problem — "this server is sending spam" — and it reads the queues and the logs across the machine and comes back with the specific compromised mailbox and the evidence. It investigates; you decide. Nothing it proposes touches a server until you approve the exact change. That asymmetry — free to investigate everything, gated on every action — is deliberate, and we'll come back to it.

One honest note up front. This is our product and we're not going to pretend otherwise or bury it in coyness. But everything described here is a real way of working, not a feature list — the principles hold whether you run our software or wire the equivalent together yourself from osquery, Prometheus, and a pile of scripts. We think CentralHost is the better answer, and the rest of the book is us showing our work. Read it and decide.

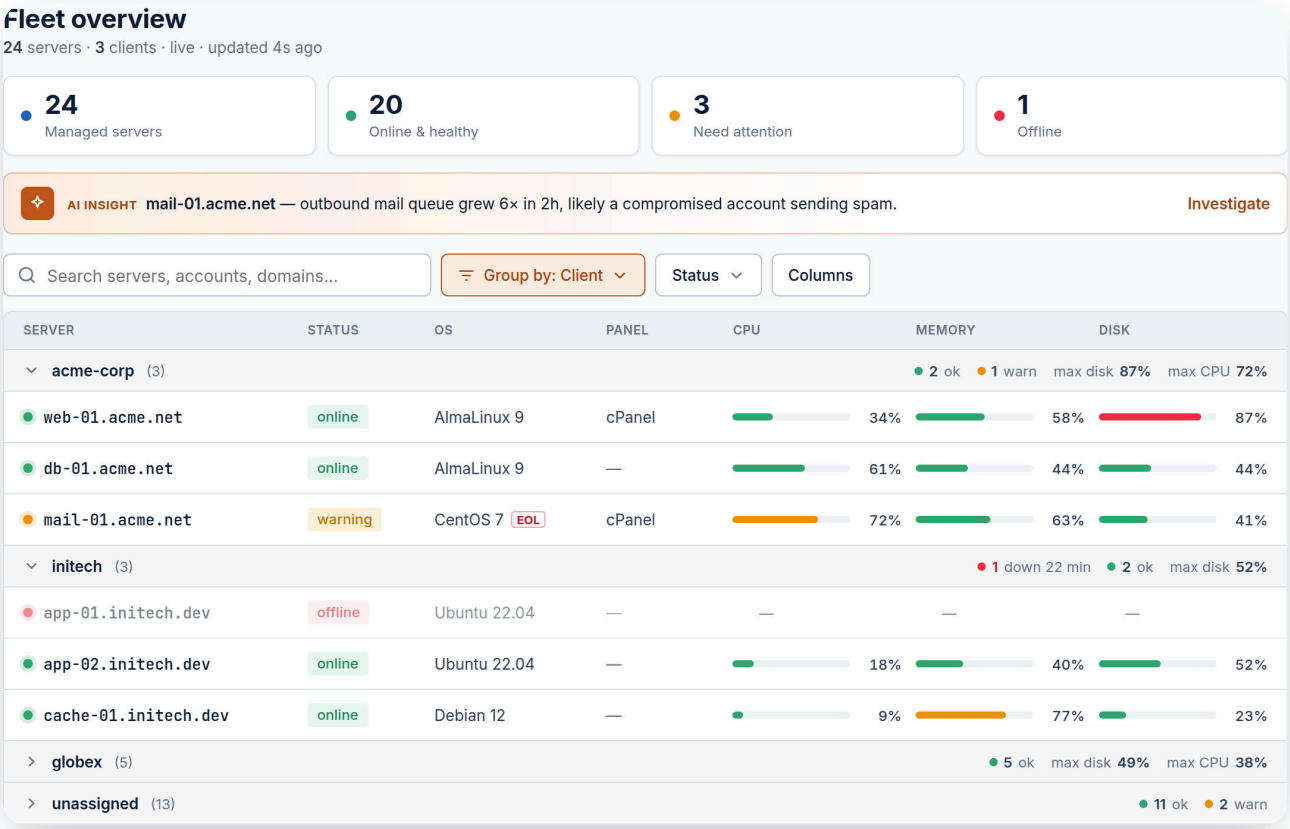
Part 1 — One place for everything

Everything CentralHost does rests on one thing being true: that it always knows, accurately and right now, what you have. Monitoring, alerts, the firewall, the assistant — none of it works against a fleet you can't see clearly. So that's where the tour starts. Before you can take care of forty servers, you have to be able to look at them.

See your whole fleet, current to the minute

You install one small agent on each server. That's the whole setup — no central database to populate by hand, no spreadsheet to seed. From the moment the agent comes up, the server starts describing itself: what operating system and version it runs, what's installed, what's listening on which port, how much disk is left, what's running. You don't enter any of it. The machine reports it, and keeps reporting it, so the picture stays current on its own.

The result is a single view of every server you own, each row reflecting what's true on the box this minute — not what someone remembered to type last quarter.



Every server in one place, describing itself — status, OS, and load reported live by the agent, not maintained by hand.

This is the end of the spreadsheet, stated plainly. The inventory stops being something you maintain and becomes something you simply have. And because it's discovered rather than typed, it can answer the questions a frozen list never could: which servers are still on an end-of-life OS, which are exposing a port they shouldn't, which haven't taken an update in two months. Those stop being an afternoon of SSH-ing around and become a filter.

Across every provider, in one view

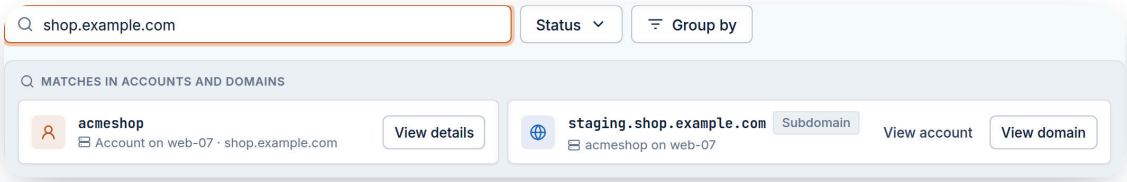
Here's the catch that breaks every provider's own dashboard: your servers don't all live in one place. Some are on Hetzner, some on DigitalOcean, a couple on whatever was cheapest that month. Each provider gives you a console that sees only its own machines, and the moment your fleet spans two of them, no single one of those consoles can show you your fleet. You're left doing the integration in your head, tab by tab.

CentralHost sits above the providers. The agent reports to it regardless of where the server runs, so a box on Hetzner and a box on DigitalOcean land in the same list, look the same, and are managed the same. There's no "Hetzner section" and "DigitalOcean section" — there's just your fleet, in one view, whoever happens to be renting you the hardware. The provider becomes an implementation detail instead of a boundary you have to work around.

Organized the way you actually think

A flat list of forty servers is still forty things. You don't think in hostnames — you think in *clients* ("Acme's boxes"), in *environments* ("is that the staging one?"), in *roles* ("all the mail servers"). The list only becomes navigable when you can collapse it into the groups that match how you actually reason about your fleet.

So you tag the things only a human knows — which client owns a server, whether it's production or staging, what it's for — and you *filter* on the things the machine already reports. "Everything running PHP 7" or "all the Ubuntu 20 boxes" aren't tags you maintain; they're filters over discovered facts, always current, free. Between the two, almost any group you'd ever want to act on is one click away.



"Acme's production boxes still on the old OS" is a filter, not a memory exercise — human tags and discovered facts combined.

That's what turns *managing servers* into *managing a fleet*: the moment you stop addressing machines one at a time and start addressing meaningful groups of them. Everything in the rest of this book — watching, protecting, patching, acting — is done against these groups. Which is why we built the seeing first.

With the fleet in view, the next question is the obvious one: how do you know, without staring at it all day, when something on one of these machines starts to go wrong?

Part 2 — Know what's happening

Seeing the fleet is the foundation. The next thing you want from it is to be *told* when something's wrong, so you're not the one checking. That's two halves, and they fail in opposite ways when they're done badly: monitoring that quietly records the disaster nobody was watching, and alerts so noisy you've trained yourself to ignore them. The point of both is the same — to spend your attention only when it's actually needed.

Active monitoring: metrics that watch themselves

A graph is only useful if someone is looking at it. That's the quiet flaw in most monitoring: it's *passive*. It faithfully records that a disk filled up at 2 a.m. and waits patiently for you to come look in the morning, by which point the site has been down for six hours. A dashboard you have to remember to check is just a prettier spreadsheet.

CentralHost monitors actively. Every server reports its vitals continuously — CPU, memory, disk and the things that actually take a box down but rarely show up on a basic dashboard: connection tracking tables filling, file descriptors exhausting, inodes running out. You get the live graphs when you want them, per server and across the fleet. But the point isn't the graphs. The point is that you don't have to watch them, because the system watches them for you and speaks up when something crosses the line.

That's the difference between *monitoring* and *being monitored*. You're not the one staring at forty dashboards hoping to catch the one that's about to tip over. The fleet tells you.

Alerts that mean something

The reason most people end up ignoring their monitoring is that it cried wolf one too many times. A tool that emails you two hundred times a day trains you, within a week, to delete its mail unread — and then it's worse than nothing, because now you have alerts *and* you don't read them.

CentralHost's alerting is built to be the opposite: quiet until it matters, and specific when it speaks. Alerts are edge-triggered — you hear when a condition *starts*, not every five minutes for as long as it lasts. They carry severity, so a disk at 95% and a disk at 99% don't arrive looking identical. There's anti-flap built in, so a metric bouncing across a threshold doesn't spray you with noise. And because the alert is wired to the same inventory as everything else, it doesn't just say "high load on 178.x.x.x" — it knows it's Acme's mail server, and what's running on it, and tells you that.

An alert you trust is one you'll actually act on. The whole design goal is to send you few enough messages that you read every one.

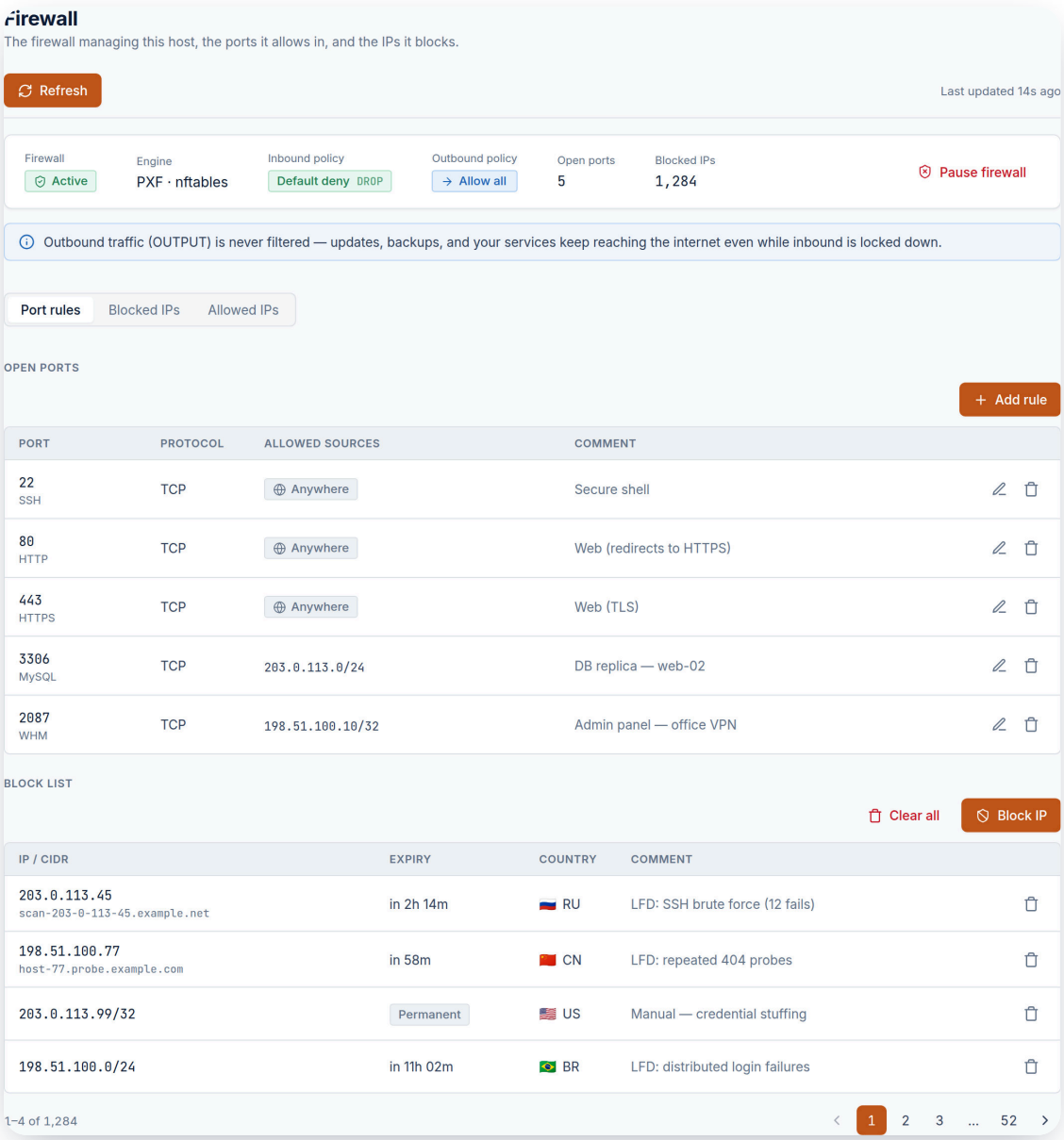
Part 3 — The admin's toolbox

This is the part of CentralHost that replaces the reflex of SSH-ing into a box to do a thing. Firewall rules, a WAF, your databases — the everyday administration you'd normally do by hand, on each server, done instead from one place, on a group when you want, and written to an audit trail.

Firewall administration, fleet-wide

Managing a firewall by hand is the kind of task that's tedious on one server and genuinely error-prone across forty — different syntaxes, the constant low-grade fear of locking yourself out, no single place to see what's actually open where.

CentralHost manages the firewall the box already runs. You see, in plain terms, which ports are open on a server, and you open or close them from the console — no SSH, no remembering which dialect this machine speaks. You keep a denylist of IPs and apply it where you need it. And two things are deliberately safe by design: the outbound channel is never cut, so the agent can't strand itself, and there's a guard against the classic mistake of closing the very port you're connected through. The firewall you already trust, with a fleet-wide view and a console in front of it.

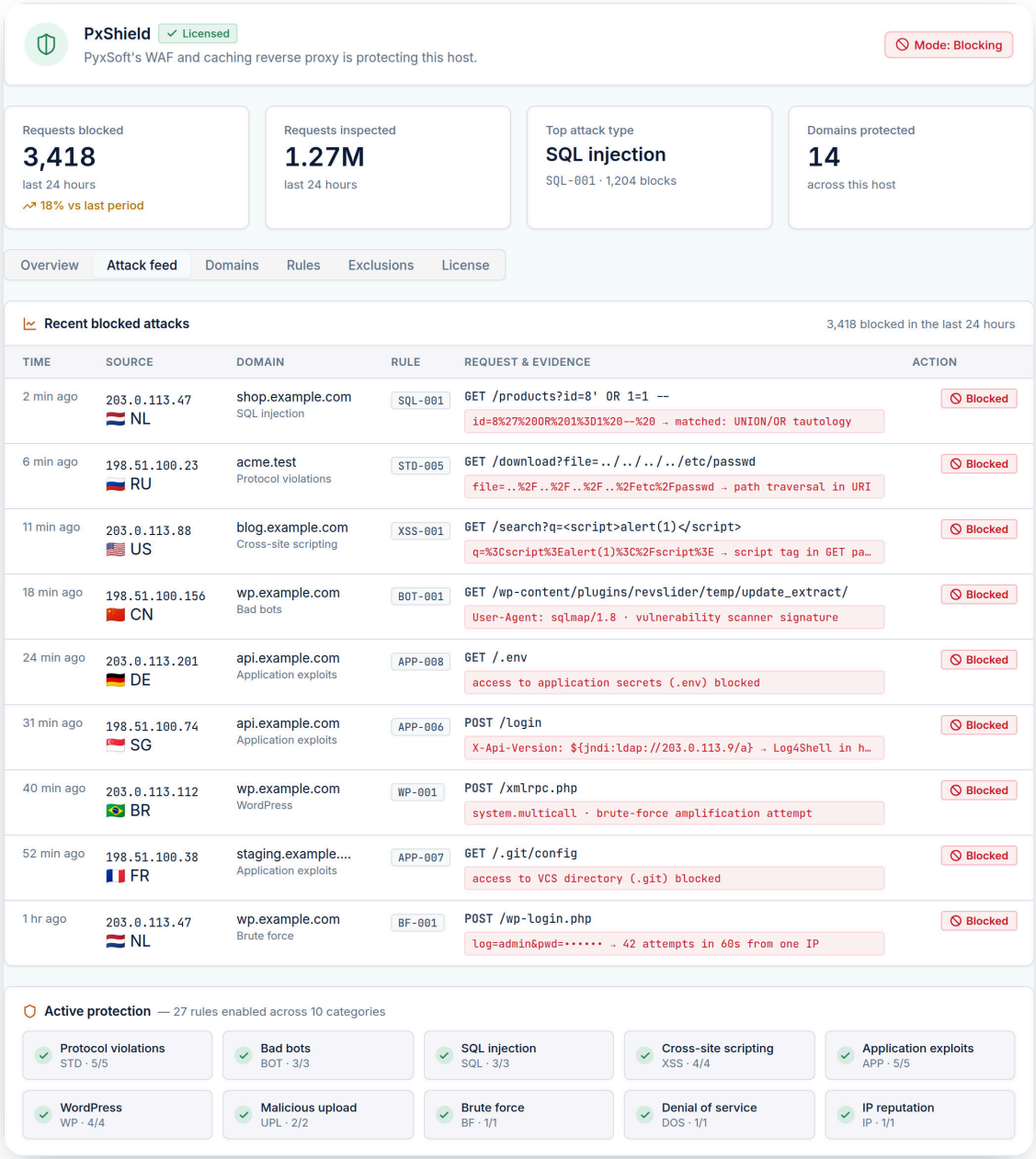


Open ports, IP denylist, and posture for the firewall the box already runs — fleet-wide, from the console, with the outbound channel never cut.

A managed WAF in front of your sites

A firewall decides who can reach a port. It does nothing about the attack that arrives over port 443 looking exactly like normal web traffic — the SQL injection, the path traversal, the probe for a known vulnerable plugin. That's the job of a web application firewall, and standing one up by hand on every server is the kind of project that never quite gets done.

CentralHost puts a managed WAF — PxShield — in front of your sites, and makes turning it on a matter of clicking rather than an afternoon. Once it's running, the attacks it blocks aren't just a number; you see the events, with the evidence of what was attempted, so a "we're under attack" gut feeling becomes something you can actually look at. And because those attack events live in the console with the rest of your fleet's data, they're available to the AI assistant in Part 4 — which means "is anyone attacking my sites, and how" becomes a question you can simply ask.

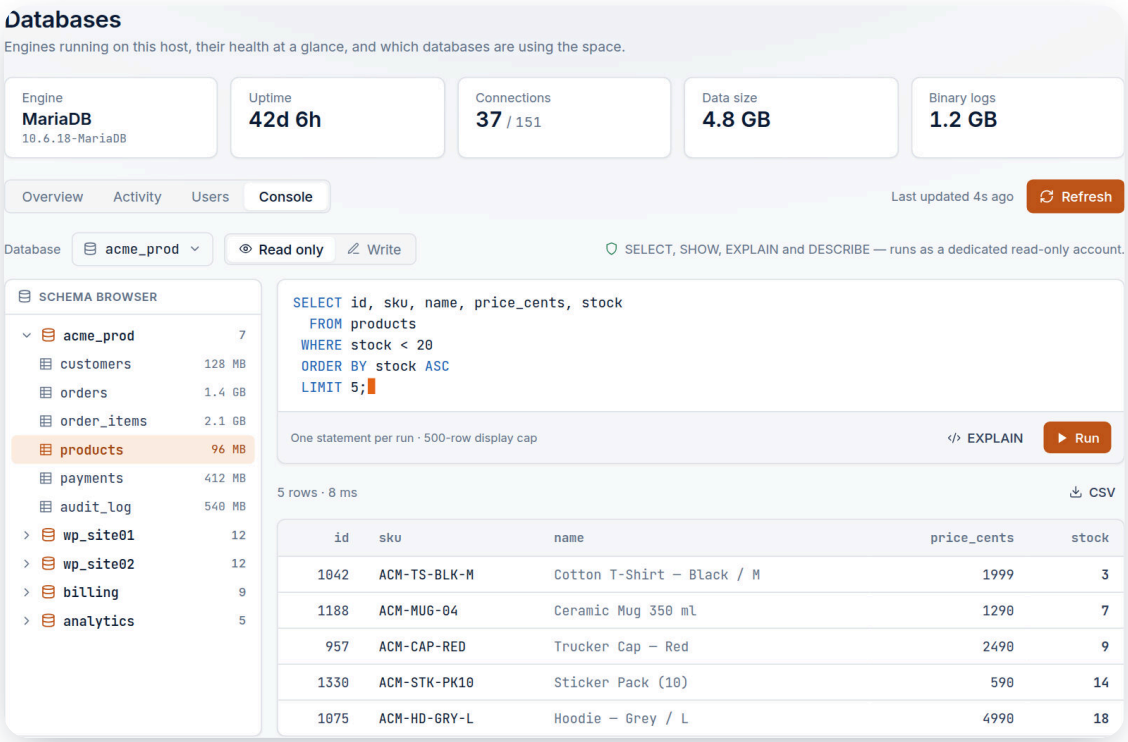


PxShield blocking attacks at the edge — each event carries the evidence of what was attempted, available to ask the assistant about.

Database administration from the console

Here's a task you'd normally solve by SSH-ing in and reaching for the `mysql` or `psql` client, or by exposing a web admin like phpMyAdmin and hoping nobody else finds it: looking after the databases on your servers. CentralHost brings that into the console.

You get your MySQL, MariaDB, and PostgreSQL databases laid out by engine — versions, sizes, the users and their privileges — discovered on demand, not stored anywhere it doesn't need to be. When a database is the problem, the live activity view shows you what's actually running on it right now, and lets you kill the one runaway query that's pinning the server, without leaving the page. And when you genuinely need to run SQL, there's a console with a schema browser built in — guarded, so it permits the reads and inspection you reach for in a hurry and refuses the operations that have no business being run from a web console. Database housekeeping, audited, without a separate tool and without opening another port to the world.



Your databases from the console: engines and sizes, users and privileges, live activity, and a guarded SQL console with a schema browser.

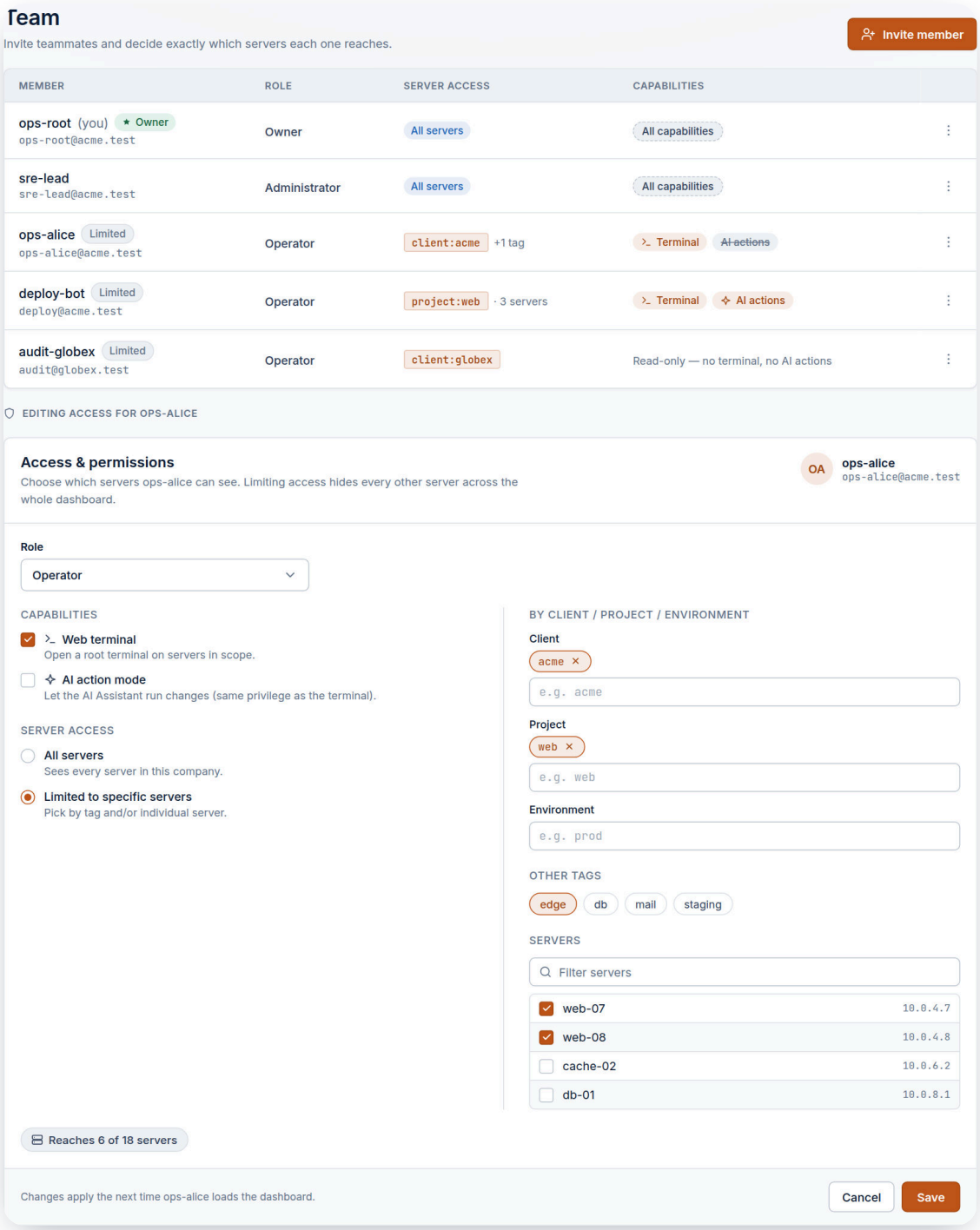
Roles and permissions: give the team exactly what they need

A fleet is rarely a one-person operation. You have an admin who runs everything, a contractor who should only touch one client's servers, a junior who can look but not change. Handing all of them the same all-or-nothing login is how mistakes and breaches happen.

CentralHost lets you invite people to your account and scope them precisely. Owners and admins see and do everything; the interesting role is **Operator**, where access is built from two independent dials:

- **Scope — which servers they see.** All of them, or just the ones that match a tag (`client:acme`, `project:web`), or an explicit list. The contractor working on Acme's sites sees Acme's boxes and nothing else — not the rest of your fleet.
- **Capabilities — what they're allowed to do.** Two switches matter most: whether they can open a root **web terminal**, and whether they can let the **AI apply changes** rather than only investigate. Everyone can read and use the assistant read-only; the powerful, irreversible privileges are granted deliberately, per person.

So "give the new contractor terminal access to Acme's three servers, but don't let the AI make changes on their behalf" is something you can actually express, in a few clicks, instead of approximating it with a shared password and good intentions. Least privilege, without the friction that usually makes people skip it.



Scope an Operator to one client's servers and grant capabilities — web terminal, AI actions — one person at a time.

Patch and harden by the group, not the box

Keeping servers patched and hardened isn't hard knowledge — everyone knows updates should be applied and SSH shouldn't be wide open. It's hard *bookkeeping*. On forty machines, the question stops being "how do I patch a server" and becomes "which of my forty haven't been patched, which are exposed, which drifted out of line while I wasn't looking" — and that's a question a spreadsheet can't answer and a per-box routine can't keep up with.

This is where the self-maintaining inventory from Part 1 pays off a second time. Because every server reports what it's running, the fleet can tell you — as a filter, not a manual audit — which boxes are behind on updates, which are still on an end-of-life OS, which are exposing a port they shouldn't, which are running the package named in today's advisory. You ask the fleet a security question and the fleet answers, across every machine at once, current to the minute.

And because Part 1 also let you group your servers, you act on the answer the same way: by the group, not the box. "Patch Acme's production web servers." "Show me everything still exposed on the old SSH config." The unit of work becomes the meaningful set — the eight that are Acme's, the three on the old OS — instead of forty individual chores you have to remember to finish. Hardening stops being a thing you do once per server and forget, and becomes a posture you can see and hold across the whole fleet. The work that used to drift the moment you looked away is now something the inventory keeps honest for you.

That covers protection. What's left is the thing you reach for when protection isn't enough and you actually have to get in.

Part 4 — Act, safely

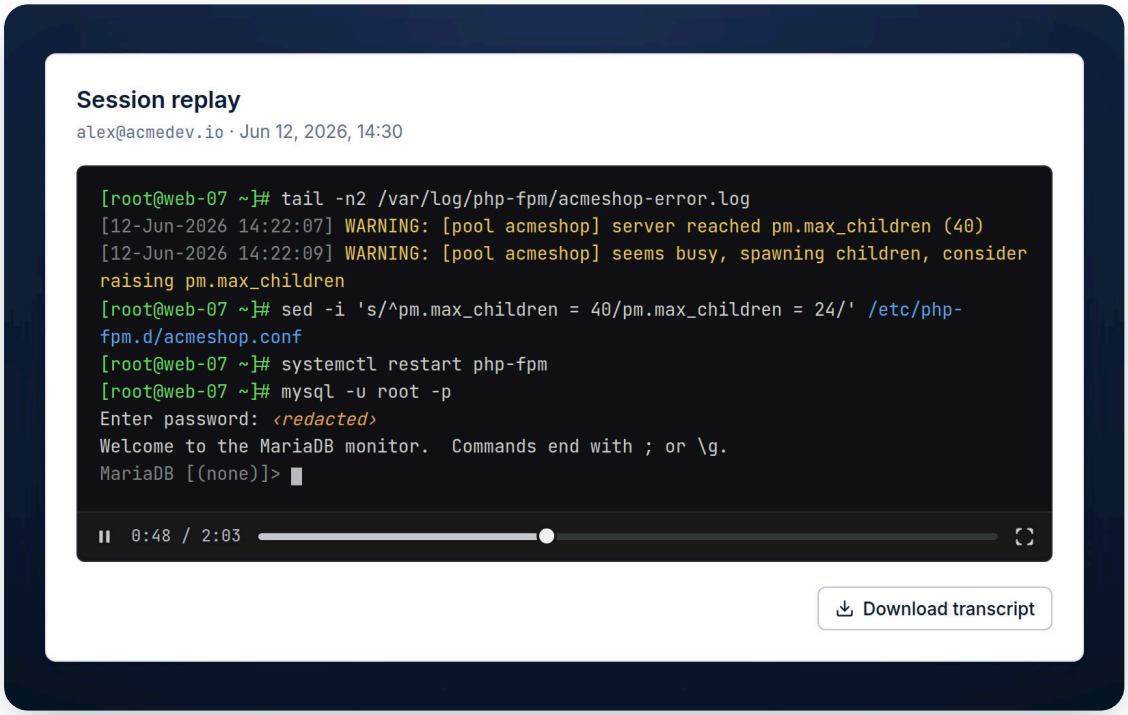
Seeing the fleet, knowing when something's wrong, keeping it protected — all of that is worth a great deal, and most of it you do without ever touching a server. But sometimes you have to act. Something is broken and you need to get in and fix it, or figure out *why* in the first place. CentralHost gives you a spectrum of ways to do that, from fully hands-on to fully delegated: a terminal where it's all you, the same terminal with an AI copilot reading over your shoulder, and an assistant that does the whole investigation for you. You pick how much to hand off, problem by problem. What stays constant across all three is the safety rail: every irreversible action waits for your hand on the switch.

A terminal one click away, fully audited

When you do need a shell, you shouldn't have to go hunting for the right key, remember which jump host fronts this client, or — worst of all — leave SSH open to the internet just so you can get in when something breaks.

CentralHost gives you a terminal in the browser, one click from the server you're already looking at. It runs through the agent, which means you don't expose SSH to do it: port 22 can stay closed to the world while you still get a full, interactive shell on demand. No key juggling, no bastion to configure by hand, no standing exposure.

And because it runs through the platform, every session is accountable. Sessions are recorded and replayable — who connected, when, and exactly what they did, available afterward in the audit trail. That matters the moment you're managing other people's servers: "we got in and fixed it" becomes something you can show, not just say. The same machinery powers time-boxed guest access, so you can hand a single audited terminal to an outside contractor or a vendor's support engineer without giving them a credential or a standing door.



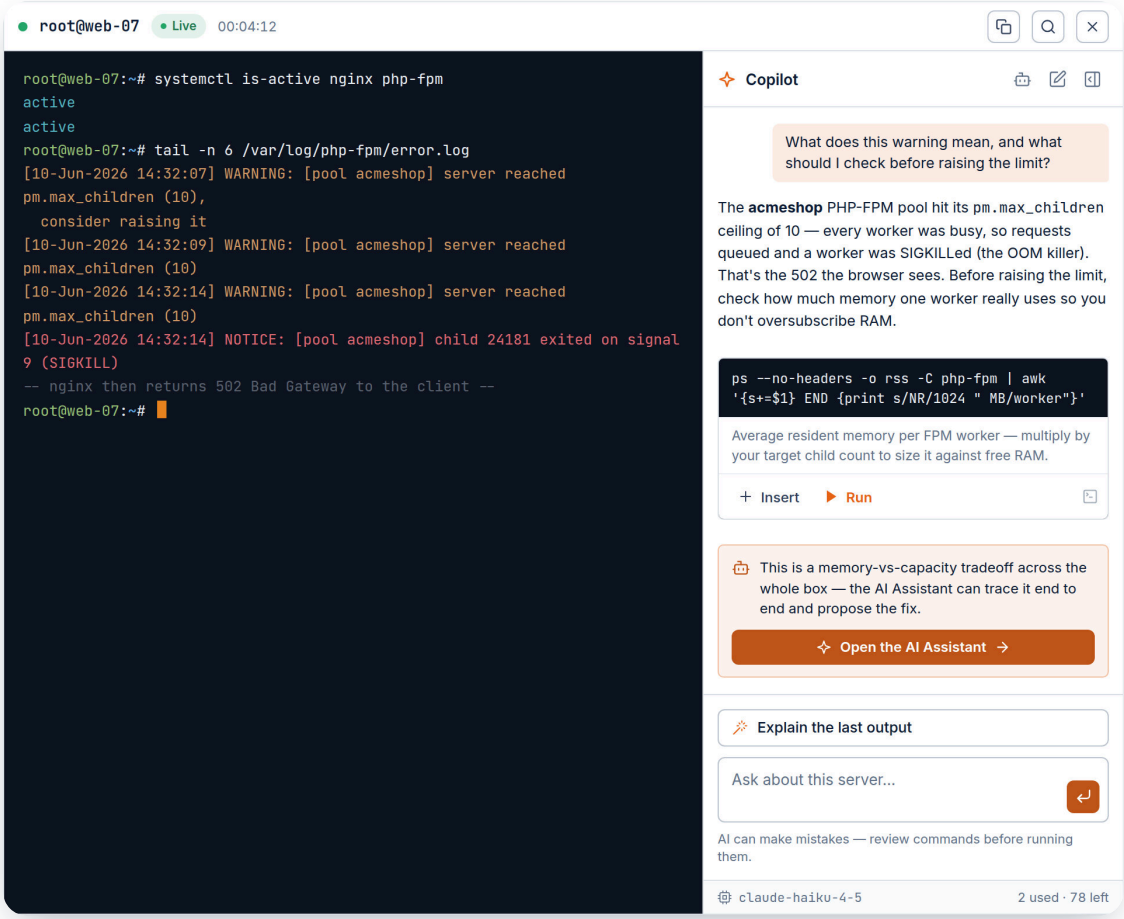
Every terminal session is recorded and replayable — what was done, by whom, and when. SSH stays closed to the world; access runs through the agent.

That's the fully hands-on end of the spectrum: your hands, on a shell, with a safety net under them. But you don't have to be alone down there — the same terminal can read over your shoulder.

The same terminal, with a copilot

Here's the move everyone makes when a command throws something they don't recognize: copy the line, alt-tab to a browser, paste it into a search box or a chatbot, read three answers that don't quite fit, alt-tab back, and try to remember what you were doing. The fix was never the slow part. The context-switch was.

CentralHost's web terminal has a copilot docked in a rail right beside the live shell. It reads what's already on your screen — when you ask a question, it carries a snapshot of your recent output with it — so you don't have to describe anything. nginx is throwing 502s and the PHP-FPM log is full of `pm.max_children`? You don't explain that; the copilot already read it. There's a one-click *Explain the last output* for the most common move of all: something just happened and you want to know what it means before you touch anything.



The copilot reads the same screen you do — explains the error in plain language and hands you the next command. You're still the one who runs it.

When it suggests a command, it doesn't make you retype it. Each suggestion is a card with two buttons: *Insert* types it into your prompt so you can edit it and press Enter yourself, *Run* sends it to the shell. Either way **you** execute — the copilot never runs anything on its own. And anything destructive (`rm`, `dd`, `mkfs`, dropping a database, rewriting firewall rules) is flagged and won't go on the first click; it arms a confirm step you have to deliberately approve. The fast path stays fast; the irreversible path makes you stop and look.

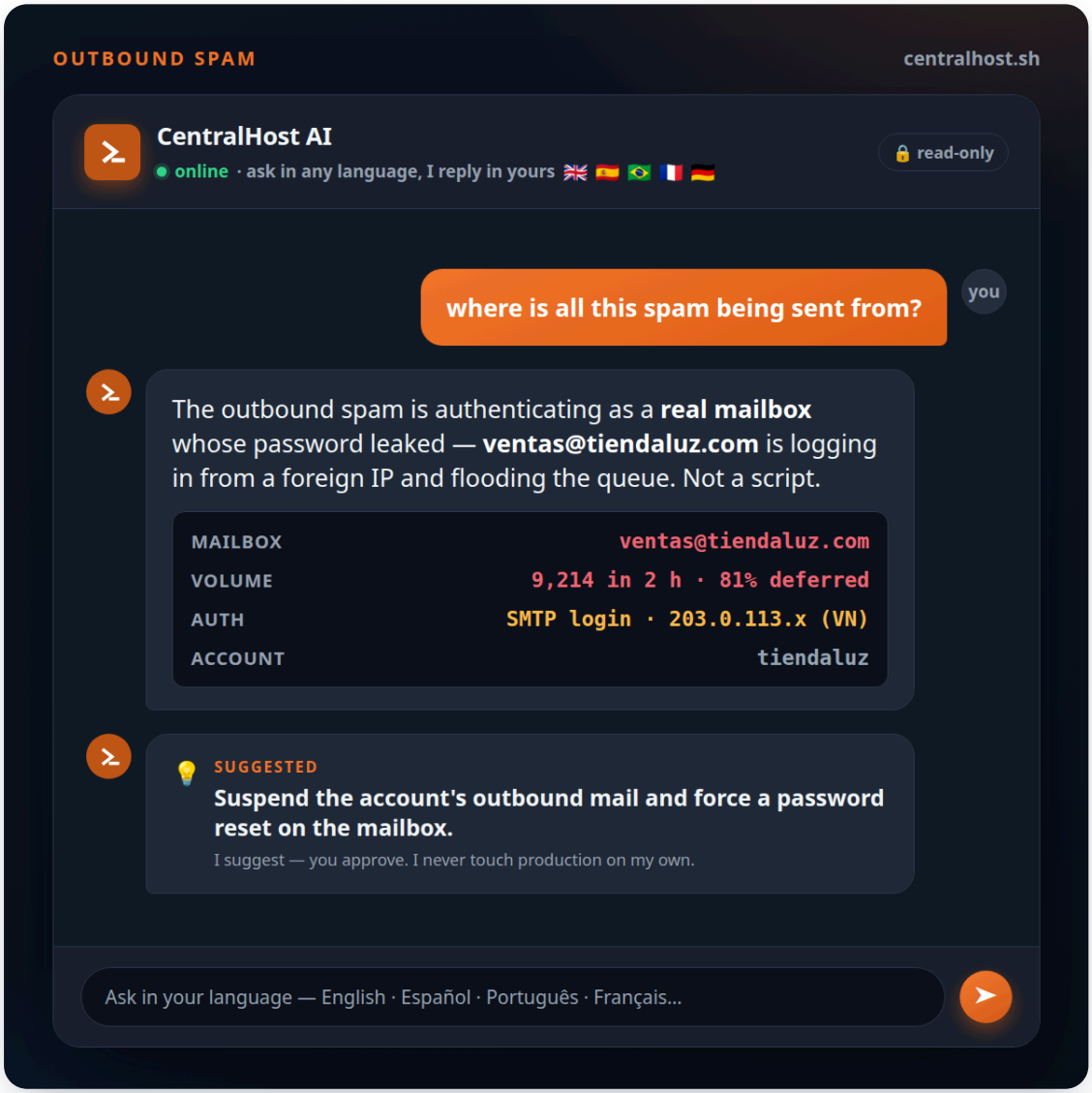
It's deliberately the *light* tool — quick, single-shot, on a cheap fast model by default, with the cost shown in the status line so a quick question stays a quick question. And it knows its limits: when something is bigger than the shell in front of you — a root-cause trace across logs, metrics, and services — it offers to hand the question to the AI Assistant, pre-filled, with one click. Which is the other end of the spectrum.

The AI assistant that investigates for you

This is the feature the whole product is pointed at, and it's worth saying plainly what it is and what it isn't.

The hardest part of running a fleet was never knowing how to fix a server — you already know that. It's the *walk*: when something breaks on one of forty machines, finding which machine, which process, which line in which log is a slow, manual slog that gets longer with every server you add. The assistant does that walk for you.

You talk to it in plain language. *"This server is sending spam." "Why did the nightly backup fail?" "Which of my sites is someone attacking?"* It goes and looks — reading the inventory, the metrics, the running processes, the logs, the mail queue, the firewall state, the WAF's attack events, everything CentralHost already has in one place — and comes back with an answer and the evidence behind it. Not "here are some graphs, good luck," but the specific compromised mailbox, the specific failing job, the specific exposed port.



"This server is sending spam." The assistant reads the queue and logs across the box and comes back with the exact compromised mailbox — the tedious walk, done for you.

That example isn't hypothetical — it's a real investigation the assistant has run. What would have been an hour of manual log-reading came back in the time it takes to read the answer. The knowledge required was ordinary; the *looking* was the cost, and the looking is exactly what it took off your plate.

Now the part that makes it safe to put on a production fleet. **The assistant investigates freely and acts only with your permission.** Reading is unrestricted — it can look at anything, because the worst case of a wrong reading is a wrong answer you catch. But the moment it wants to *change* something — edit a config, restart a service, run a command as root — it stops and asks. You approve the specific change, on the specific machine, right then; nothing destructive fires until a human who understands the consequence says so. You can scope that approval (just this once, this session, this server), and for bigger jobs it can lay out a plan for you to approve as a block before it touches anything. Every action it takes is audited like any other.

That asymmetry — free to investigate, gated on every action — isn't a limitation we'll loosen later. It's the correct shape for the tool, and the reason you can hand it a real problem on a real server without holding your breath. (The Appendix makes the longer argument for why.)

It runs on the leading models — you choose, per investigation, between faster and cheaper or slower and more thorough — billed by what you actually use. But the model is the engine, not the point. The point is that the part of the job that never scaled with how good you are — paying full attention to forty machines at once — is finally something you can hand off.

This is what makes CentralHost more than a nicer dashboard. A dashboard shows you the fleet. This investigates it.

Closing · Start with one machine

Step back and look at what the tour covered. You can see your whole fleet in one place, current to the minute, across every provider, organized the way you actually think. You know when something's wrong because the fleet tells you, instead of you watching dashboards. You can manage the firewall, a WAF, and your databases without SSH-ing into anything. You can patch and harden by the group instead of the box. And when you do need to act, you have a terminal with a copilot reading over your shoulder and an assistant that will do the whole investigation for you — both with the same rail under them, where nothing irreversible happens without your hand on the switch.

That's the difference between running servers and running a fleet. Not a bigger spreadsheet — a different way of working, where the machines describe themselves, the watching is done for you, and the tedious walk across forty boxes is something you can hand off.

Here's the honest part. You don't have to believe any of this from a book. The whole thing is built to be tried on one machine, with nothing to migrate and nothing to commit to: install the agent on a single server and watch it fill itself in. The inventory you've been maintaining by hand appears on its own, current, in a couple of minutes. From there you add the second server, and the third, and at some point — usually sooner than you'd expect — the spreadsheet is just closed, and you don't reopen it.

Start with one machine. See if it tells you something you didn't know.

Ready when you are. — **www.centralhost.sh**

Appendix · System Administration in the Age of AI

Every few years something arrives that is supposed to make system administrators obsolete. Configuration management was going to do it. Then the cloud. Then containers, then "NoOps," then serverless. None of them did. What each one actually did was move the work — up a layer, into a different shape, onto a larger number of machines. The job didn't disappear; it got broader and more abstract, and the people who adapted ended up responsible for more than they ever were before.

AI belongs in that lineage, and it deserves the same unsentimental reading. It is not going to replace you. But it is going to move your work again, and it is worth being precise about where.

The bottleneck was never knowledge

Here is the thing the "AI replaces sysadmins" framing gets wrong: it assumes the hard part of the job is knowing things. It isn't, and it hasn't been for a long time. A competent administrator already knows what a load average means, how to read an Exim log, where Postfix queues mail, what a SYN flood looks like. That knowledge is not the scarce resource.

The scarce resource is **attention applied across machines**. The bottleneck is that when something goes wrong on one of forty servers, finding *which* server, and *which* process, and *which* line in *which* log, is a manual walk. You SSH in. You `tail` something. You don't find it. You SSH into the next one. The knowledge of what to look for is instant; the labor of looking is what eats the afternoon. At one server, that walk is trivial. At forty, spread across three providers, it is the entire job — and it is the part that doesn't scale with how good you are.

This is the right way to understand where AI fits. Not as a replacement for the knowing, but as a replacement for the **walking**.

What this kind of work is actually good at

Large language models are genuinely, unusually good at a specific shape of task that happens to be most of incident triage: take a large, messy, semi-structured pile of text — logs, process lists, config files, mail queues — correlate across it, and narrow a wide space of possibilities down to a small one. That is not a parlor trick. It is the exact motion a senior administrator performs in their head, except the model can do it across forty machines at once without getting tired on the thirty-first.

The honest framing is that AI is a **first responder, not a decision-maker**. It is the thing that runs toward the problem, gathers everything relevant, eliminates the eighty percent that's irrelevant, and hands you a short list with its reasoning attached. It compresses the walk. What it produces is not a verdict you act on blindly; it is a dramatically smaller problem than the one you started with.

A concrete example, because this should not stay abstract. A server is sending outbound spam — the kind of incident that gets an IP blacklisted and a hosting account suspended. The classic version of solving this is an hour of work: figure out which account, read the mail logs, find the compromised mailbox or the leaked SMTP credential, confirm it. The knowledge required is ordinary. The *looking* is the cost. Point an assistant at that same server and the work it does is exactly the walk — read the queue, correlate senders, find the one mailbox the volume is coming from — and what it returns is the specific compromised mailbox, with the evidence. That is not the model being clever. It is the model

doing the tedious, mechanical, attention-bound part that a human is slow at not because they lack skill but because they have one pair of eyes.

What you do not delegate

It would be dishonest to write this as a celebration. The same capability that makes an assistant useful makes it dangerous in exactly one direction: **acting**. Reading is safe. A tool that reads your logs and tells you what it found, at worst, is wrong and wastes your time. A tool that *changes* your servers — edits a config, restarts a service, deletes a file, runs a command as root — is a different category of risk entirely, and pretending otherwise is how people get burned.

So the line is not "AI or no AI." The line runs straight through the middle of the work:

- **Investigation is safe to hand over.** Reading, correlating, diagnosing, summarizing. The worst case is a wrong answer you catch.
- **Mutation stays gated.** Anything that writes to a server should require a human to look at the specific change and approve it. Not approve "the AI" in general — approve *this command, on this machine, right now*. The model proposes; you dispose. The destructive verb does not fire until a person who understands the consequence says so.

This is not a temporary safety rail to be removed once the technology matures. It is the correct permanent architecture. The value of automating the investigation is enormous and the risk is low. The value of automating the *decision to act destructively* is small and the risk is unbounded. CentralHost's assistant reflects that asymmetry in its design, not just in its marketing — it is read-only by default, and every mutating action is a deliberate, audited, human-approved step.

Why the fleet is where this matters most

All of this is mildly useful on one server and transformative across forty, for the same reason the rest of this book exists: the problems that break administrators are problems of *scale*, not of *depth*. You can hold one server in your head. You cannot hold forty. The walk that AI compresses is precisely the walk that gets longer with every machine you add — and it gets longer faster than linearly, because the machines are heterogeneous, on different providers, running different stacks, drifting in different directions.

An assistant that can investigate across the whole fleet, that already has the inventory and the metrics and the access in one place, is not doing forty times one server's worth of work. It is doing the thing a human fundamentally cannot: paying full attention to all of them at once. That is the part of the job that does not scale with talent, and it is exactly the part worth handing over.

The shape of the job after this

The administrator does not go away. The administrator moves up — again, as they have every other time. Less of the day spent on the mechanical walk: the SSH-in, the tail, the grep, the next machine. More of it spent on the parts that were always the actual job and never the bottleneck — deciding what to do, judging whether the diagnosis is right, owning the change, designing the system so the incident doesn't recur.

That is not a smaller job. It is a better one. The most valuable thing a senior administrator brings has never been the ability to read a log file faster than the next person. It is judgment — knowing which

problems matter, which fixes are safe, when the obvious answer is wrong. AI does not threaten that. It clears away the toil that was crowding it out.

The administrators who struggle with this shift will be the ones who mistook the toil for the job. The ones who thrive will be the ones who were always frustrated that the toil got in the way of the work. For them, this is the tool they've been waiting for — provided it knows its place: investigate everything, decide nothing, and never touch a server without being told to.