Connect your tools

Claude Code, Cursor, GitHub Copilot, Gemini CLI, Codex CLI.

- Overview
- Install the Endpoint Suite
- Govern all detected tools
- Claude Code
- Cursor
- GitHub Copilot
- Gemini CLI and Codex CLI
- Check that it worked

Overview

Connect your tools

The Sentilai Endpoint app configures your AI tools to route through the Gateway. On a governed machine, open the app and use **Govern all detected tools** — everything below happens automatically. This page documents what each adapter actually does, for transparency and troubleshooting.

Claude Code

The adapter points `ANTHROPIC_BASE_URL` at the Gateway (session-wide), so Claude Code's own credential — your Claude subscription session or API key — passes through unchanged. Claude Code reads this at startup: fully quit and reopen it after connecting.

Cursor

Cursor is routed via its custom endpoint mechanism, which can only carry an API key — so Cursor runs in **managed key mode**: the Gateway injects your organization's provider key from the Provider Vault. Cursor picks the change up on a full restart (the app offers a one-click restart).

GitHub Copilot

- **Copilot CLI**: configured via environment (`COPILOT_PROVIDER_BASE_URL` + a device credential). Managed key mode.
- **Copilot in VS Code**: a guided card in the app — VS Code's BYOK settings need one manual paste (VS Code offers no API for another extension to do it). Requests still transit GitHub's servers; routing adds governance on the model call.

Gemini CLI

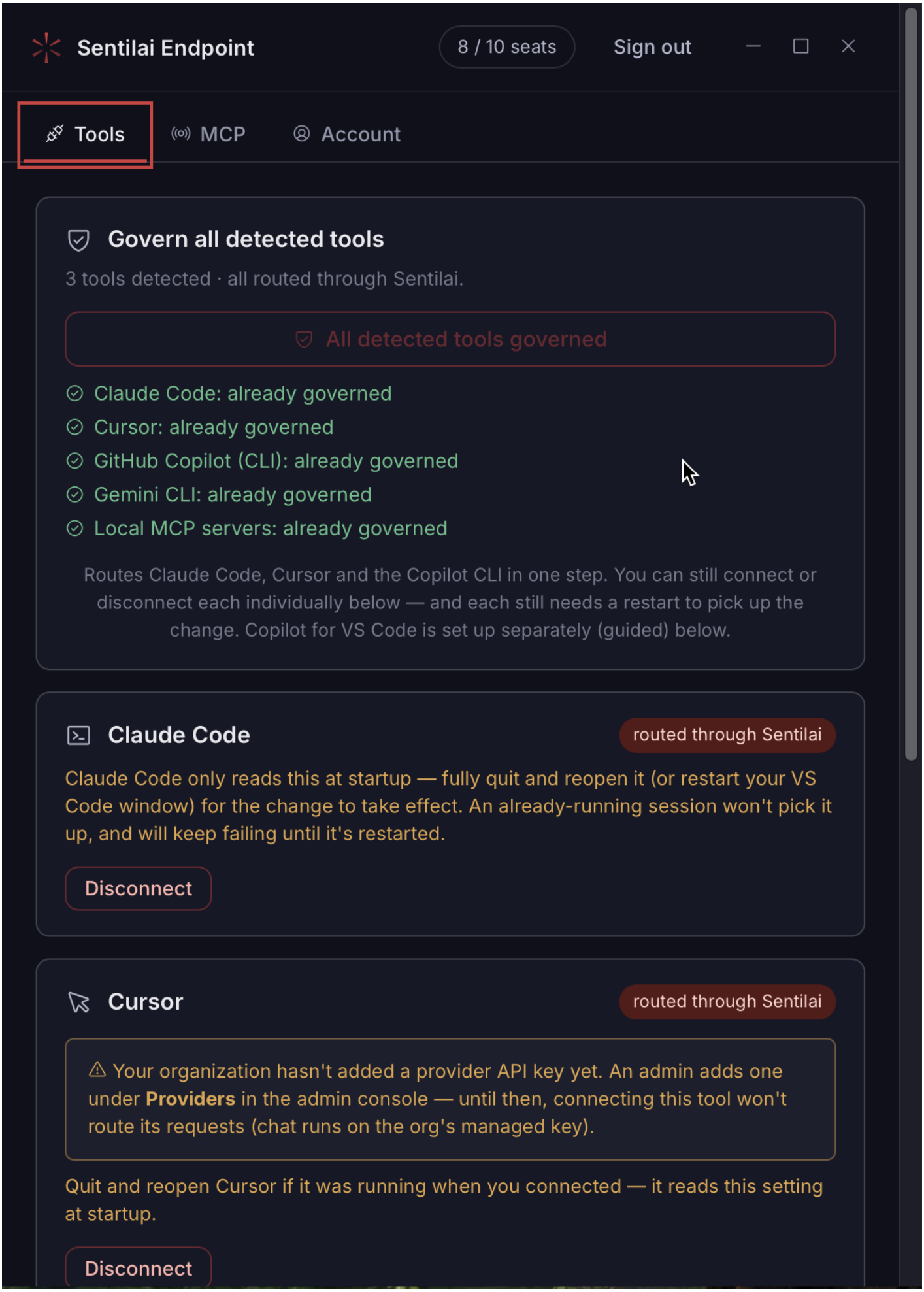
Configured via `GOOGLE_GEMINI_BASE_URL` + `GEMINI_API_KEY` (the device credential). Managed key mode — the Gateway injects your organization's Gemini key upstream.

Codex CLI

The adapter adds a managed provider block to `~/.codex/config.toml` pointing at the Gateway, reversible byte-for-byte on disconnect. Managed key mode with your organization's OpenAI key.

Verifying

Every governed tool shows **connected** in the app, and its traffic appears on the Activity screen within seconds of use.


Sentilai Endpoint
8 / 10 seats
Sign out
Tools
MCP
Account
Govern all detected tools
3 tools detected · all routed through Sentilai.
All detected tools governed
Claude Code: already governed
Cursor: already governed
GitHub Copilot (CLI): already governed
Gemini CLI: already governed
Local MCP servers: already governed
Routes Claude Code, Cursor and the Copilot CLI in one step. You can still connect or disconnect each individually below — and each still needs a restart to pick up the change. Copilot for VS Code is set up separately (guided) below.
Claude Code
routed through Sentilai
Claude Code only reads this at startup — fully quit and reopen it (or restart your VS Code window) for the change to take effect. An already-running session won't pick it up, and will keep failing until it's restarted.
Disconnect
Cursor
routed through Sentilai
⚠ Your organization hasn't added a provider API key yet. An admin adds one under Providers in the admin console — until then, connecting this tool won't route its requests (chat runs on the org's managed key).
Quit and reopen Cursor if it was running when you connected — it reads this setting at startup.
Disconnect

Everything the Suite found on this machine, and whether each one is routed through Sentilai yet.

Install the Endpoint Suite

The Endpoint Suite is the desktop app that puts a developer's machine under governance. It signs them in, registers the device, and points their AI tools at the Gateway.

Installers for both platforms are at **sentilai.com/download** — the page shows the current version, file sizes and SHA-256 checksums.

macOS

1. Download the `.dmg` from sentilai.com/download (or the link your admin gave you).
2. **Drag the app into Applications before opening it.** Opening it from the mounted disk image leaves macOS running it from a temporary read-only location, and it will not be able to store its device key.
3. Open it from Applications. The app is signed and notarized by Apple. On first open, macOS still shows its standard confirmation for anything downloaded from the internet — the dialog says *"Apple checked it for malicious software and none was detected"*. Click **Open**. That is the only prompt; there is no "unidentified developer" warning.

The download runs on both Apple Silicon and Intel Macs (one universal app).

Windows

1. Download `setup.exe` and run it. It installs **per user** — no administrator prompt, nothing written to Program Files.
2. Windows SmartScreen may show a "Windows protected your PC" warning. The installer **is** Authenticode-signed — click **More info** and check that the publisher reads **BRIGHTOPS SMART SOLUTIONS SOCIEDAD LIMITADA** (Sentilai's legal entity), then **Run anyway**. The warning reflects SmartScreen's *reputation* score, which builds as more machines install the app; it disappears on its own over time. If **More info** shows no publisher at all, stop — that file did not come from us.

Signing in

The app opens your normal browser to sign in with a passkey — Touch ID, Windows Hello, or a security key. **Nothing is typed into the app itself**, which is why there is no password field anywhere in it.

If the browser window closes without finishing, press **Sign in with browser** again.

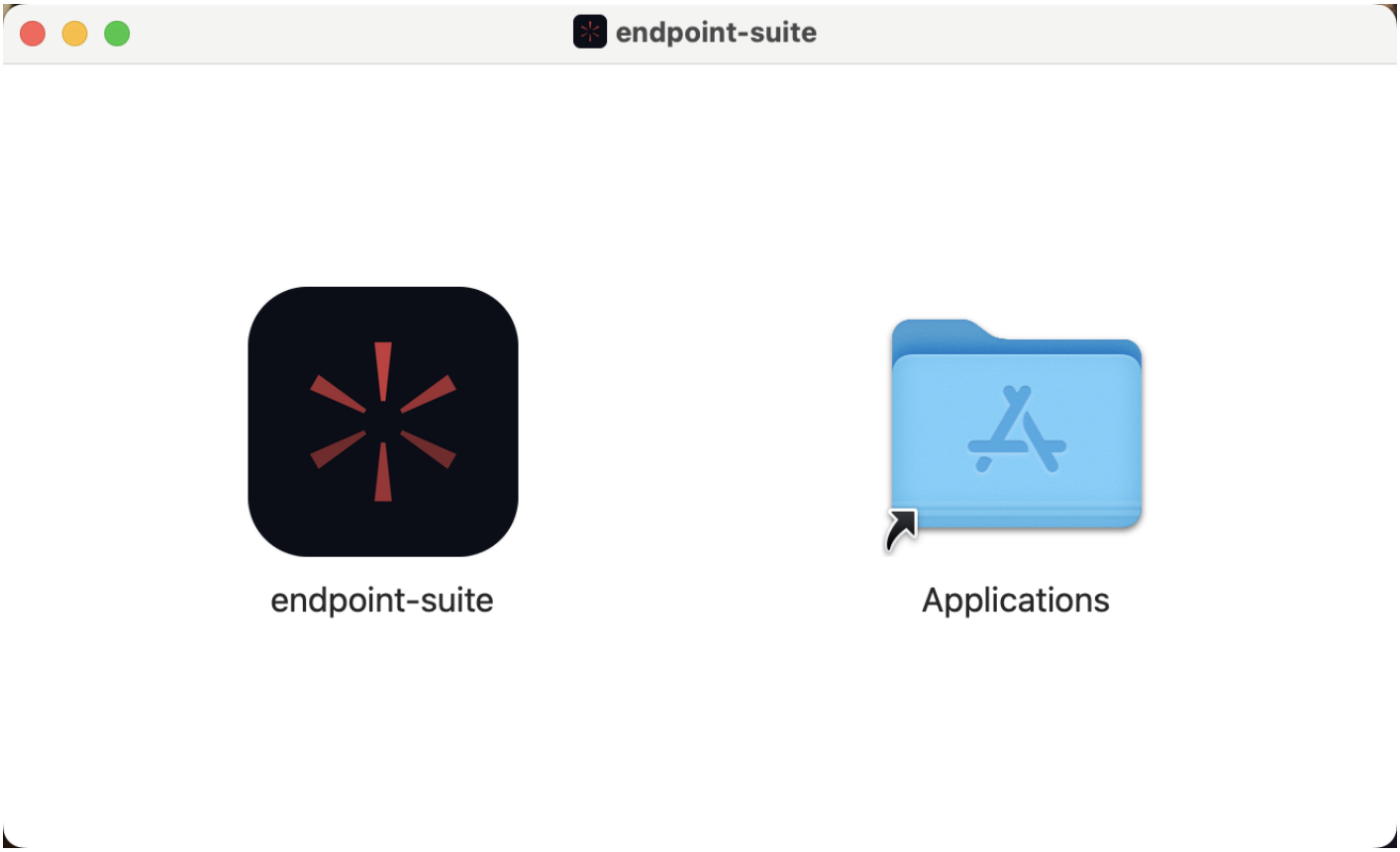
Device identity

On first run the app creates an ECDSA P-256 key pair and stores the private key in the operating system's secure storage — Keychain on macOS, Credential Manager on Windows. The private key never leaves the machine and is not something you can export or copy to another computer.

The card on the sign-in screen shows whether this succeeded. If it says it failed, see Endpoint Suite problems.

Seats

Signing in consumes one of your organization's seats. The topbar and the **Account** tab both show the current count.



The disk image opens on the one instruction that matters: drag the app into Applications. Run from the image instead and macOS executes it from a translocated read-only path, where it cannot keep its device key.

Govern all detected tools

The fastest way to put a machine under governance: one button, every tool it can find.

Doing it

Open the **Tools** tab. The card at the top says how many AI tools were detected and how many are not yet routed through Sentilai. Press **Govern all detected tools**.

The app then reports, per tool, whether it was connected, was already governed, or failed — with the reason.

Then restart the tools

This is the step people miss. Every one of these tools reads its configuration once, at startup. Until you fully quit and reopen it — not just close the window — it keeps talking directly to the AI provider and nothing appears in Activity.

On macOS that means quitting the application properly (?Q), not closing its window. For command-line tools, close and reopen the terminal.

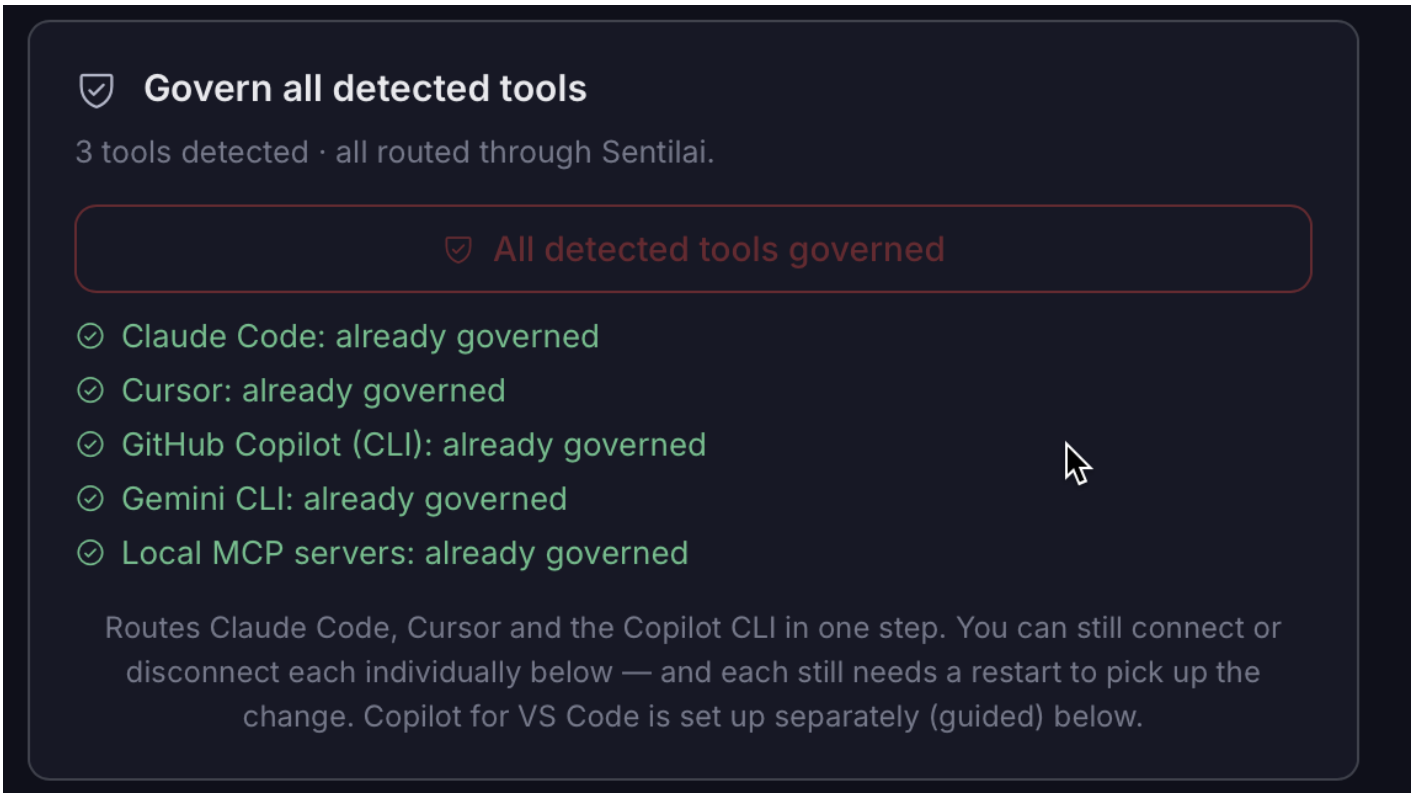
What it does not cover

- **GitHub Copilot inside VS Code** needs a separate manual setup, because VS Code's custom-endpoint feature cannot be configured from outside. See GitHub Copilot.
- Tools that are not installed are listed as **not detected** and skipped.

Automatic governing

The **Account** tab has an **Automatic governing** toggle, on by default. With it on, any AI tool or local MCP server that appears later is governed in the background without anyone pressing anything — and any tool whose configuration drifts back to the direct provider gets re-governed.

If your organization deploys a managed configuration, this toggle may be pinned on and shown as "managed by your organization".



One button routes every detected tool at once — and names each one it has already governed.

Claude Code

Claude Code is the smoothest of the adapters, and the only one where a subscription can carry through unchanged.

Before you connect

Open Claude Code's own account panel and check it is using your **Claude subscription** rather than an API key. If it says "API Key", the request is already being billed somewhere else and connecting it to Sentilai will not change that.

Connecting

Tools ? Claude Code ? Connect. The app explains what it is about to edit and asks you to confirm twice, because it is modifying a settings file that Claude Code owns.

Your subscription keeps billing exactly as before. Sentilai governs and audits the request on the way through; it does not become the payer.

Restart it

Claude Code reads this setting only at startup. **Fully quit and reopen it** — the app says so in bold, and it is still the most common reason someone reports "I connected it but nothing shows up in Activity".

Disconnecting

Disconnect restores the previous configuration. If your organization has locked this down through a managed configuration, the button is not shown and the card explains why.

What it looks like when it works

The tool card shows **routed through Sentilai**, and the first prompt you send produces a row in the admin console's Activity feed within a few seconds.



Claude Code, routed through Sentilai. It reads this only at startup, so an already-running session must be restarted.

Cursor

Cursor can be governed, with four caveats worth knowing before you start. The app lists all four in the confirmation dialog; they are repeated here because they change what you should expect.

The four caveats

1. **Chat models only.** Cursor's Tab autocomplete keeps using its own built-in model and is not routed through Sentilai. Autocomplete traffic will not appear in Activity.
2. **Requests still transit Cursor's servers.** Cursor's architecture sends requests through its own infrastructure before they reach the provider endpoint you configure. Governance applies, but Cursor is still in the path.
3. **Cursor must be fully quit** before connecting or disconnecting — it holds its settings database open and a running instance will overwrite the change.
4. **Managed API key only.** Cursor's custom-endpoint mechanism cannot carry a subscription login, so this tool requires an OpenAI provider key on your organization.

Connecting

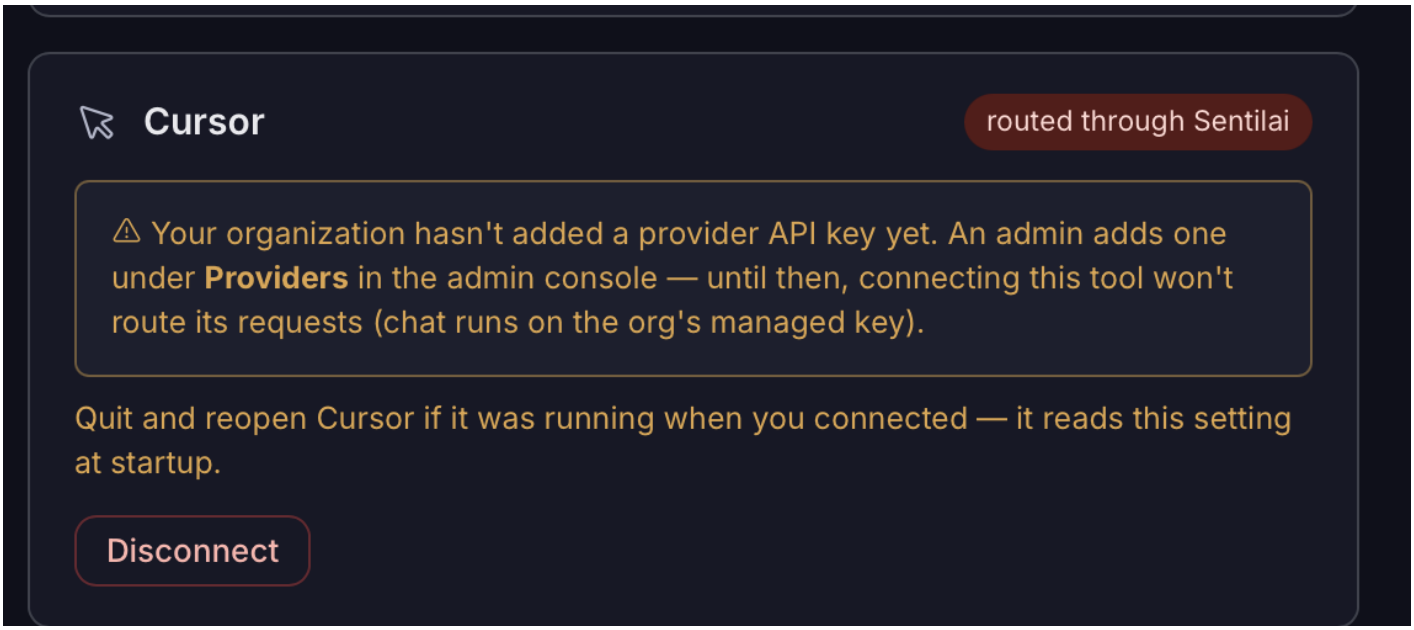
Quit Cursor. Then **Tools ? Cursor ? Connect** and confirm. The app takes a backup of the settings database before editing it, and offers a **Restart Cursor now** button afterwards.

After reconnecting

Open Cursor's **Settings ? Models** and make sure the OpenAI key is enabled and a `gpt-…` model is selected. Cursor will otherwise quietly fall back to its own models and you will see nothing in Activity.

Disconnecting

Quit Cursor first, then **Disconnect**. The saved backup is restored.



Cursor's card. The warning is real: without a provider key on the organization's Providers screen, connecting routes nothing.

GitHub Copilot

Copilot has two separate paths, because the CLI and the VS Code extension are configured in completely different ways.

Copilot CLI — automatic

Tools ? GitHub Copilot ? Connect. Fully automatic, like the other adapters. Restart your terminal afterwards so the new environment is picked up.

Copilot in VS Code — guided, manual

VS Code's custom-endpoint feature can only be configured from inside VS Code, so the app gives you a token and a short checklist instead of doing it for you.

Press **Set up VS Code**. The app copies a token to your clipboard, then:

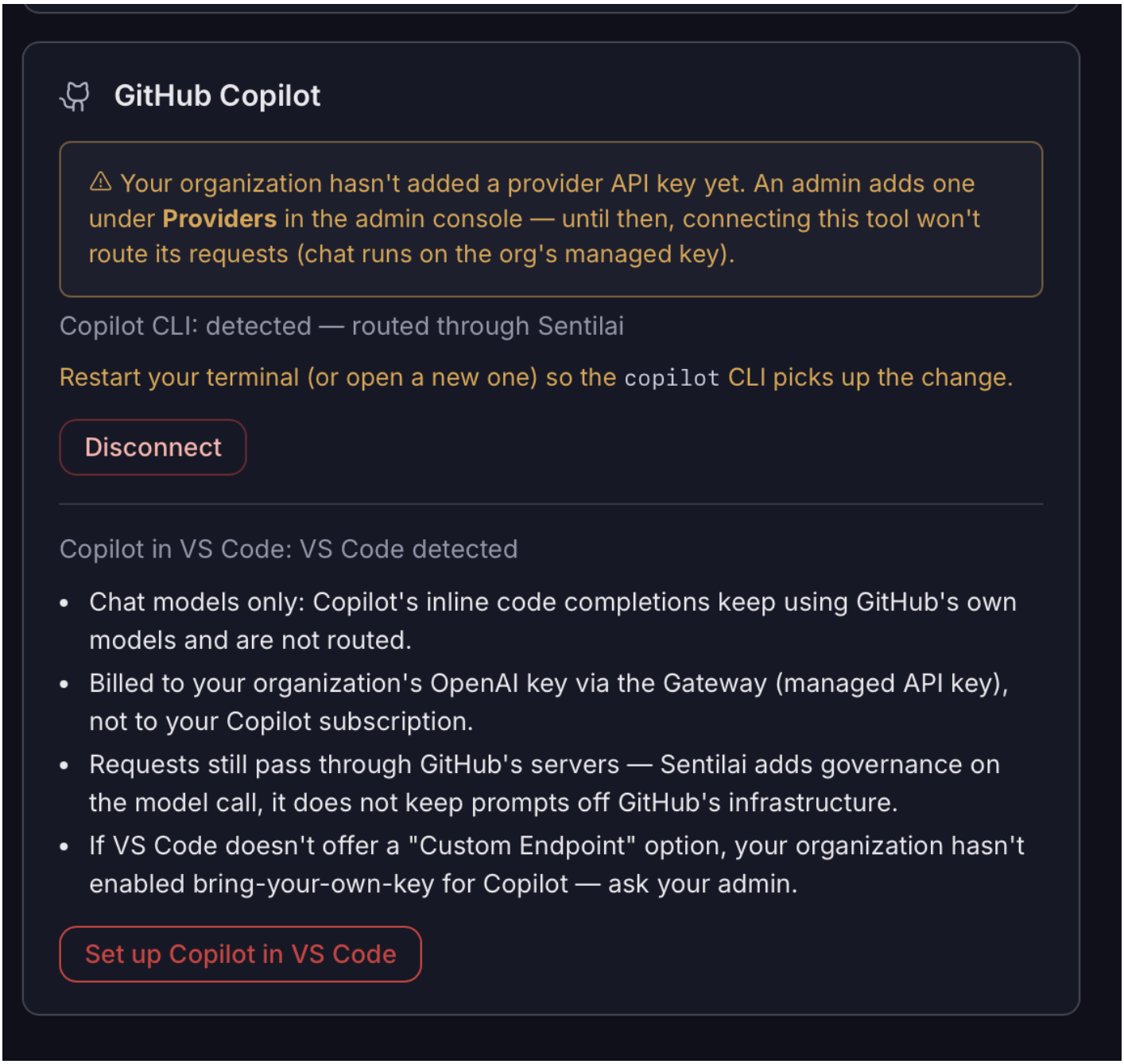
1. Command Palette ? **Chat: Manage Language Models** ? **Add** ? **Custom Endpoint**.
2. **Base URL**: the Gateway URL shown in the app.
3. Paste the API key from your clipboard (**Copy token again** if you lost it).
4. **Model**: `gpt-4o` · **API type**: `chat-completions`.

Caveats

- **Chat only.** Inline completions keep using GitHub's own path.
- Requests are **billed to your organization's OpenAI key**, not to your Copilot subscription.
- Requests **still transit GitHub's servers**.
- **If VS Code does not offer a "Custom Endpoint" option, your organization has not enabled BYOK for Copilot.** That is a setting on GitHub's side, not something Sentilai can change.

The VS Code companion extension

Separately, there is a small Sentilai extension for VS Code that shows, in the status bar, how many of your AI tools are governed (`Sentilai: 2/3`) and which base URL each is using. It is strictly read-only — it never signs you in and never writes a device token. Its one command, **Govern my AI tools**, just launches the Endpoint app.



GitHub Copilot is two things: the CLI, governed like any other tool, and Copilot in VS Code, which is set up separately and stays chat-only.

Gemini CLI and Codex CLI

Both are handled by the standard adapter path: detected automatically, connected by **Govern all detected tools** or individually from their card, and disconnected the same way.

Connecting

Tools, find the card, **Connect**. Then **restart your terminal** — like every other command-line tool here, they read their environment at startup.

Connection mode

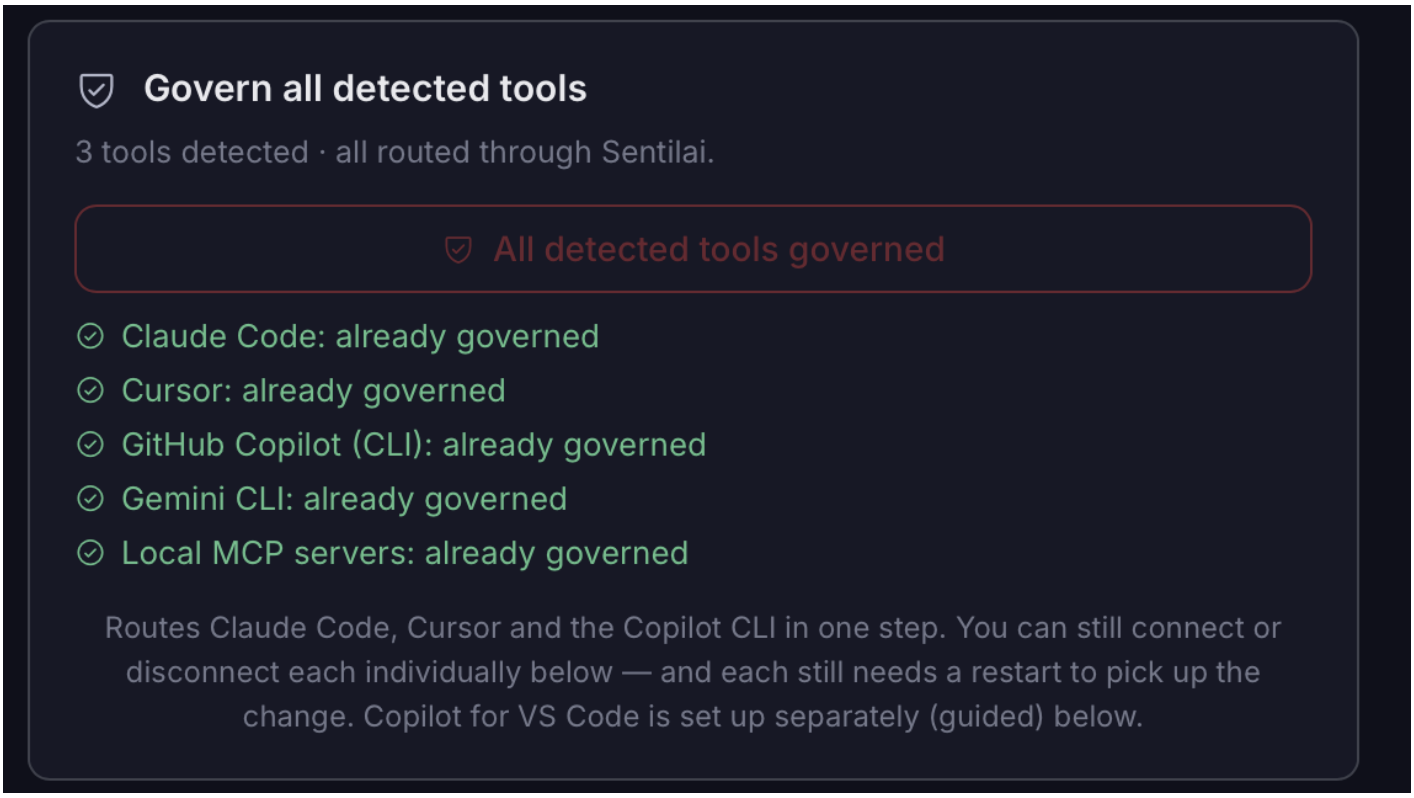
Both appear in **Policy ? Connection modes** in the admin console, where an admin chooses whether they run on the developer's own subscription or on the organization's managed provider key.

Codex CLI and posture

Codex CLI has settings that materially change how much autonomy the agent has, and the Endpoint Suite reports on them:

- `approval_policy: never` — the agent never asks before acting.
- `sandbox_mode: danger-full-access` — the sandbox is effectively off.
- `sandbox_workspace_write.network_access: true` — the agent can reach the network from inside its workspace sandbox.

These are reported, not blocked. They appear as **posture findings** on the device in the admin console's Diagnostics screen, so an admin can have a conversation about them. See Posture findings.



Gemini CLI and Codex CLI are governed through “Govern all detected tools”, which names them as it goes. Unlike Claude Code, Cursor and Copilot, neither has a card of its own on the Tools tab.

Check that it worked

Three places to look, in the order that answers the question fastest.

On the machine

The tool's card in the Endpoint Suite says **routed through Sentilai** rather than **detected**. If it still says detected, the connect step did not complete.

If you use VS Code, the Sentilai companion extension's status bar shows `Sentilai: n/m` — how many detected tools are governed.

In the tool

Send any prompt. It should behave exactly as before. If it now returns a provider error, the usual cause is managed mode without a matching provider key — see *A tool stopped working after connecting*.

In the console

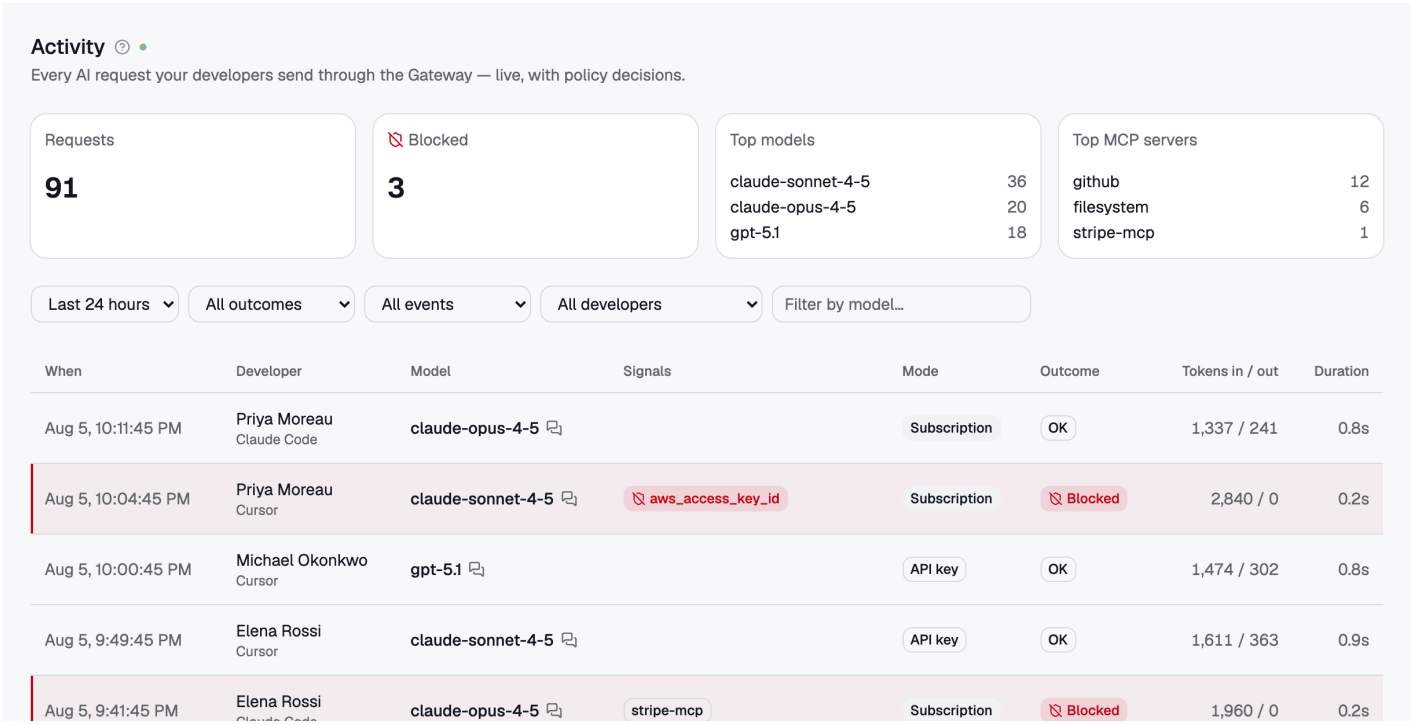
Activity, in the admin console, gets a new row within a few seconds: the developer, the tool, the model, the outcome. That row is the proof.

If the tool works but nothing appears in Activity, it is almost always the restart: the tool is still running with its old configuration. Fully quit it and try again.

If the device shows as non-compliant

The console's **Diagnostics** screen can show a **Non-compliant** badge on a device. It means a governed tool on that machine no longer points at Sentilai — usually because somebody edited the tool's configuration by hand, or the tool rewrote it.

With automatic governing on, the app repairs this by itself on its next sweep. If it keeps coming back, something on the machine is rewriting the configuration and it is worth finding out what.



The proof: your request appears in Activity within seconds, with the tool, the model and the policy decision.