Magertron™

Using Magertron Servers in Cursor

Point your editor at the servers you are entitled to call
Developer Edition

Using Magertron servers in Cursor

> **Status:** shipped · 2026-08-20 · applies to Cursor, VS Code, Claude Desktop and Windsurf

The developer portal will hand you a configuration file listing every MCP server you are entitled to reach. Save it where your editor looks for it, set a token, and the tools appear alongside everything else your assistant can do.

The whole thing takes about two minutes.

Before you start

You need three things, all of which your platform team provides.

A portal login. The portal lives at `/portal` on your Magertron address. Sign in with the same credentials you use everywhere else.

At least one server you can call. Tools you are entitled to use show a green dot — the counter at the top right of the portal tells you how many of the catalog's tools you can call. If you have none, that is a permissions question rather than a configuration one — ask whoever manages your roles.

Cursor, opened on a project folder. Cursor behaves differently depending on how you launch it. Open your project first; the rest of this assumes you have.

Get the configuration

In the portal toolbar, next to the search box, choose **Add to your IDE**.

A dialog opens showing exactly what you are about to add: the servers you can reach, with the address your editor will use. It contains nothing you are not entitled to, because the portal builds it from your own access.

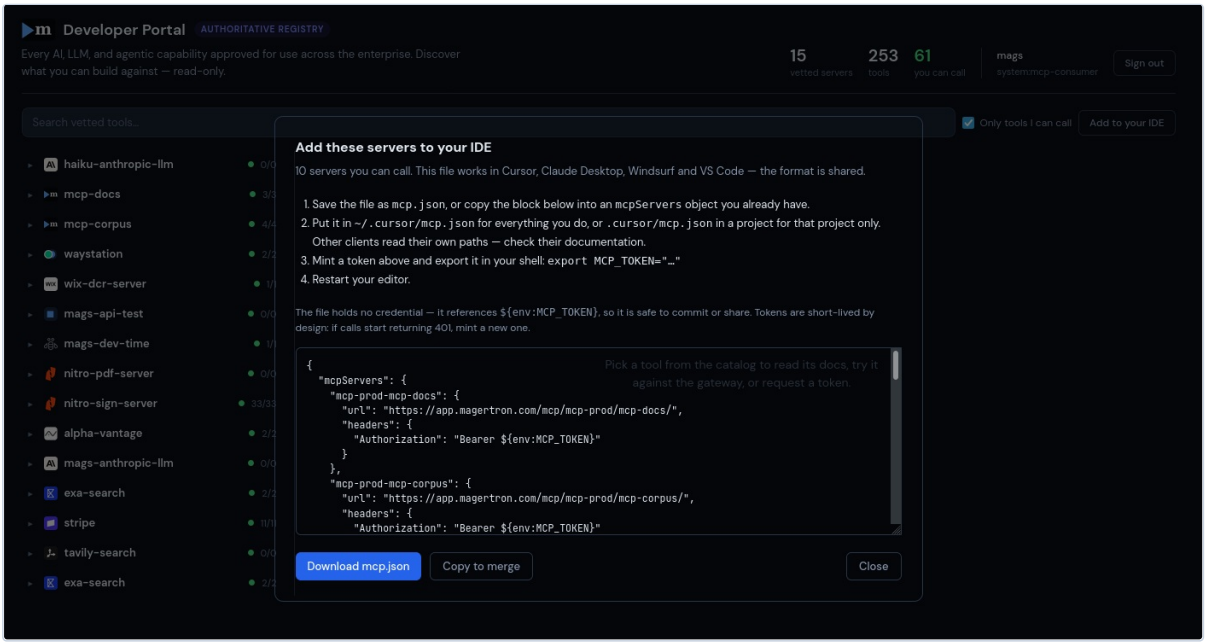

*Figure 1 — The developer portal. Green dots mark tools you can call; the **Add to your IDE** dialog builds `mcp.json` from your own entitlements and offers it for download or for merging into a configuration you already have.*

Two ways out of that dialog:

Download `mcp.json` if you have not configured MCP servers before.

Copy to merge if you have. Dropping the downloaded file over an existing configuration replaces it, and you would lose whatever was there. Copying lets you paste the entries into the `mcpServers` object you already have.

Put it where your editor looks

Cursor

Scope	Location
Everything you do	`~/.cursor/mcp.json`
One project only	`.cursor/mcp.json` in that project's root

Both work at once, so project-specific servers can sit alongside your usual ones.

Other editors read the same format at their own paths — check their documentation. The file itself needs no changes.

Set your token

The configuration references an environment variable rather than containing a credential:

```
"headers": {
  "Authorization": "Bearer ${env:MCP_TOKEN}"
}
```

> ⚠ **This is deliberate.** The file gets committed to repositories, pasted into chat and mailed around. A token in it would outlive every assumption about where it went.

Mint one in the portal — open any tool you can call and use the token panel. Give it a purpose; that label appears in the audit trail and is worth writing properly.

Then export it where your shell will find it, in `~/.zshrc`, `~/.bashrc` or equivalent:

```
export MCP_TOKEN="…"
```

> ⚠ **It must be a real environment variable.** Remote MCP servers cannot read a `.env` file — that option exists only for locally-launched servers.

Restart your editor

Cursor reads the configuration at startup. Restart it, and the servers appear under **Customize** in the sidebar with their tools listed.

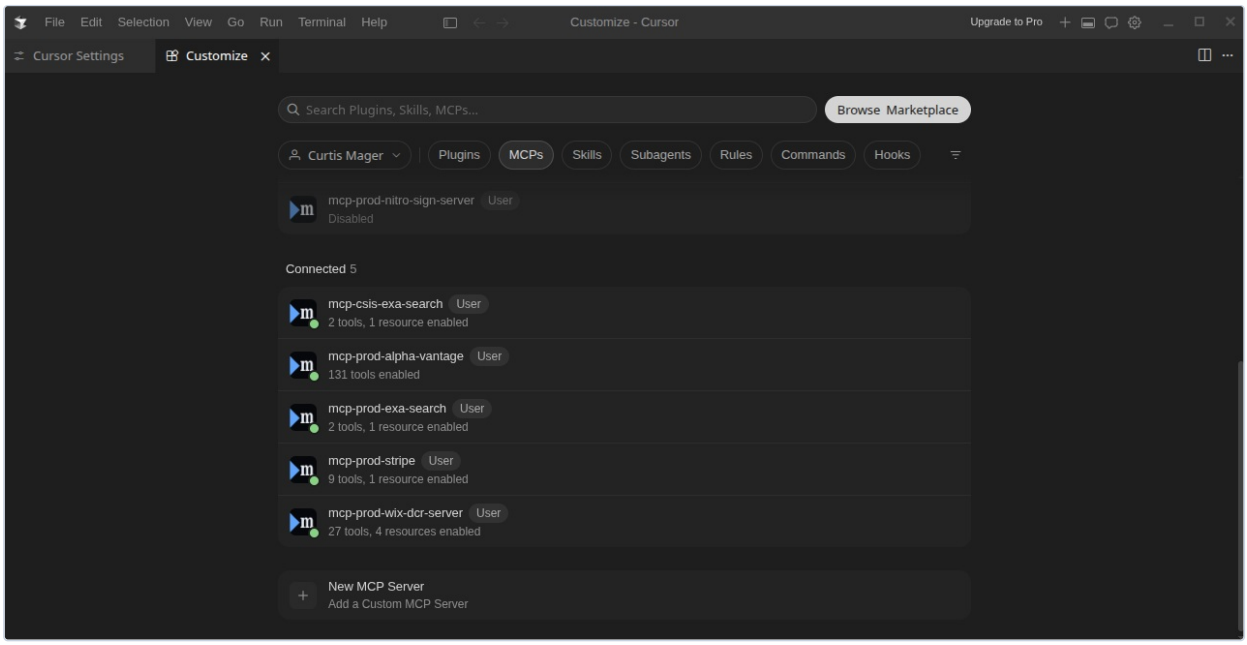

*Figure 2 — Cursor's **Customize** pane after a restart. Each entry is one Magertron server with the tools it exposes; a server you are not entitled to call shows as disabled rather than connected.*

Ask for something one of those tools does, and your assistant will offer to use it. Cursor asks before running an MCP tool, so you see the arguments first.

Tokens expire

They are short-lived on purpose. A token that lasts a year keeps working long after somebody's access has been revoked, and outlives every role change in between. Yours does not.

When calls start failing with a 401, mint a new one and update the environment variable. Restart your editor, or open a new terminal and launch it from there so it picks up the change.

If your organisation issues agent credentials through your identity provider, your platform team may have tooling that refreshes this automatically. Ask them — it is worth having.

What your platform team sees

Every call goes through the gateway, so nothing about this is private in the way a locally-launched server would be. That is the point of a governed platform, and it is worth knowing rather than discovering.

They see which tools you called, when, and the verdict on each. They do not see your prompts unless prompt recording is on, and that is a setting with its own controls.

They cannot see your token. It is shown once, in your browser, and never stored.

When it does not work

The servers do not appear after restarting. Check the file is where the editor looks — the paths above are exact. In Cursor, **Customize** in the sidebar lists what it loaded; if the file were malformed or missing, nothing would be there.

Tools appear but every call returns 401. The token is missing, expired, or the variable is not visible to the editor. Launch the editor from a terminal where `echo $MCP_TOKEN` prints something.

A tool appears but returns "denied by policy". The configuration is fine and your access is not. The server is reachable; that particular tool is not granted to your roles. The portal shows the same thing — tools without a green dot are ones you cannot call.

Everything worked yesterday and nothing works today. Almost always the token. They expire.

Submitting a server, rather than using one

This chapter is about *using* servers somebody has already approved. If you have built one and want it reviewed, that is the other direction, and it uses the Magertron extension rather than a configuration file.

> ⚠ **Open your project folder in Cursor first.** The submit action appears when you right-click a manifest **in the file tree** — with the sidebar collapsed there is nothing to right-click, and it looks as though the extension is not installed.

Installing the extension in Cursor. From a terminal:

```
cursor --install-extension magertron-submit-0.1.0.vsix
```

Download it from `https://magertron.com/cursor`. Installing through the interface works too, but Cursor's settings and extension panes differ from VS Code's and the command line is more predictable.

Restart Cursor afterwards, then set the extension's platform address. Cursor keeps these under **VS Code Settings** rather than **Cursor Settings** — search for `magertron` there.

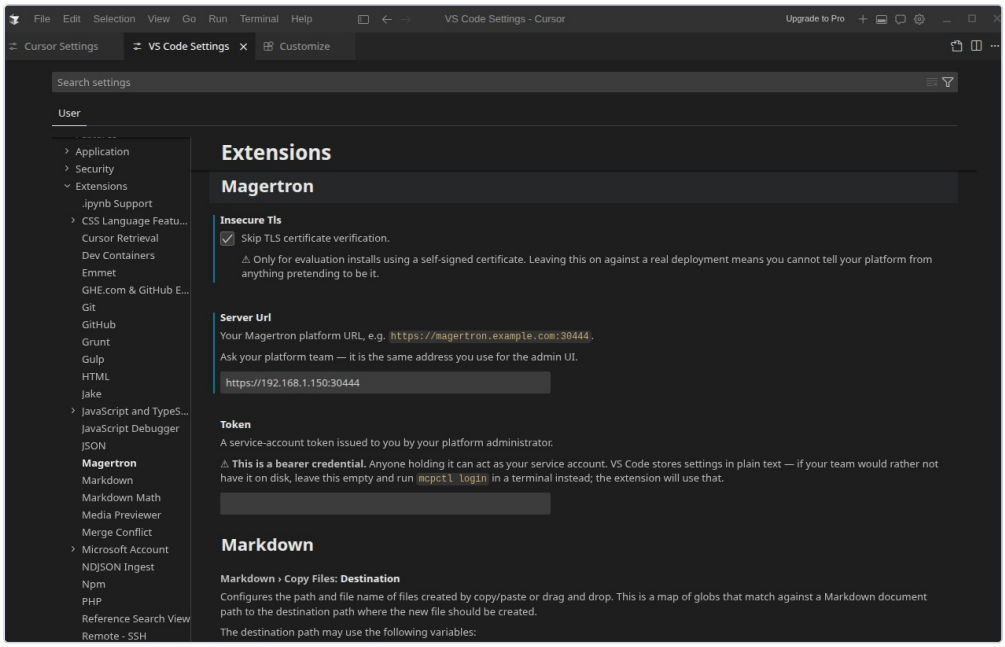

Figure 3 — The extension's three settings in Cursor, reached through **VS Code Settings**. *Server Url is the same address you use for the admin UI. Leave Token empty if this machine already has an* `mcpctl login`.

A submission from Cursor lands in the same review queue as one from VS Code, and records which editor it came from.

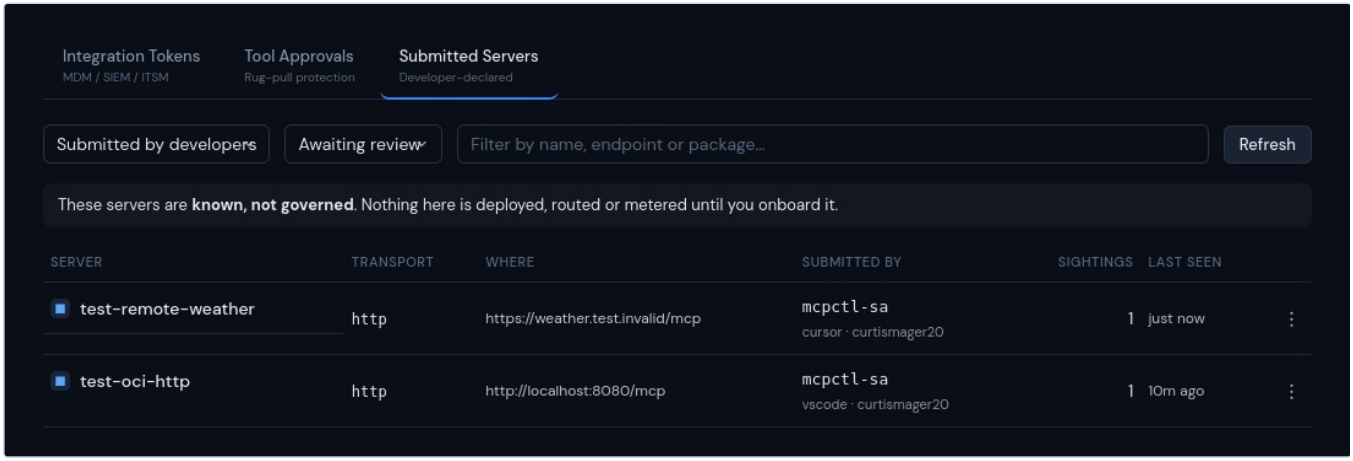

Figure 4 — The same submission as your platform team sees it, under **Submitted Servers**: *known, not governed. The tool you submitted from is recorded alongside the service account —* `cursor` *here,* `vscode` *for the entry below it.*

See *Submitting a server for review* for the rest.