**How to Set Up MCP Servers in Cursor: Step-by-Step Guide**

**MCP (Model Context Protocol)** is a trending topic among developers and AI enthusiasts. But what exactly is MCP?

**A Simple Analogy**

Imagine there are 10 people, each an expert in their own field, and they all speak different languages. If you want them to collaborate on a single task, they need a common language — like English — to communicate efficiently.

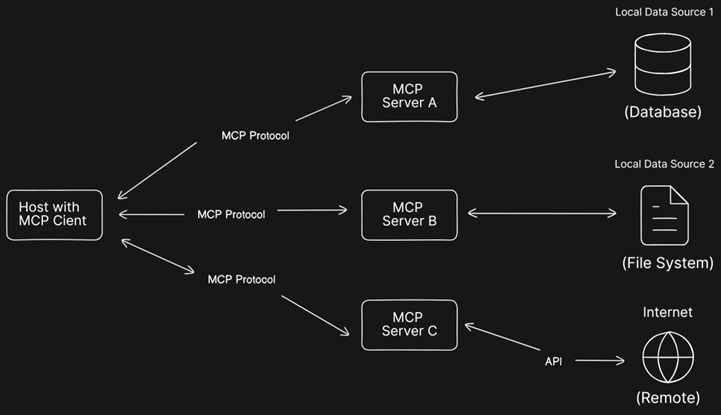
**MCP is that common language for tools and AI models.** It provides a standardized protocol that allows AI models and LLMs (Large Language Models) to connect and interact with various tools in a unified way.

**Why MCP Matters**

MCP is essential for building powerful AI agents. These agents often rely on tools to complete complex tasks. Before MCP, integrating these tools required deep coding knowledge and time-consuming setup.

Now, with MCP:

* Tool integration is easier and faster.
* Developers can focus more on logic and less on boilerplate code.



**Setting Up MCP Servers in Cursor**

In this guide, I’ll walk you through setting up MCP servers inside **Cursor**, which conveniently acts as both the **MCP Host** and **Client**.

Understanding MCP Components

* **MCP Host**: Where the request originates (e.g., Cursor IDE).
* **MCP Client**: Routes requests from the host to MCP servers and returns structured responses.
* **MCP Server**: Responds to the request and interfaces with local or remote data sources (like GitHub APIs).

In our case:

* **Cursor** = MCP Host + MCP Client
* **MCP Server** = A tool we configure, like a GitHub integration server

👉 You can find pre-built MCP servers here: [**Model Context Protocol GitHub**](https://github.com/modelcontextprotocol/servers)

**Prerequisites**

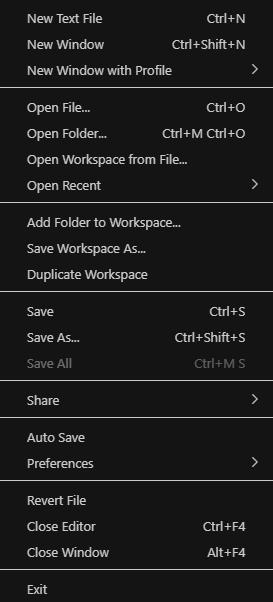
Make sure you have the following installed:

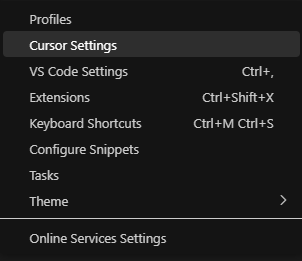
* [Node.js](https://nodejs.org/)
* [Cursor IDE](https://www.cursor.so/)
* [Docker](https://www.docker.com/)

**Step-by-Step: Adding a GitHub MCP Server**

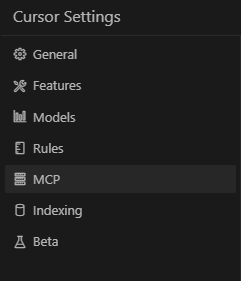
1. **Open Cursor IDE**
2. Click the **File** tab and go to **Preferences** → **Cursor Settings**







1. Navigate to the **MCP** section and click: **"Add new global MCP server"**





1. This will open the mcp.json file — this is where you’ll define your custom MCP server.

Create a GitHub Personal Access Token

Follow this guide: [Create a GitHub Token](https://docs.github.com/en/authentication/keeping-your-account-and-data-secure/managing-your-personal-access-tokens)

Paste the Following JSON into mcp.json

Add you Github Personal Access Token:

{

"mcp": {

"servers": {

"github": {

"command": "docker",

"args": [

"run",

"-i",

"--rm",

"-e",

"GITHUB\_PERSONAL\_ACCESS\_TOKEN",

"ghcr.io/github/github-mcp-server"

],

"env": {

"GITHUB\_PERSONAL\_ACCESS\_TOKEN": "<GITHUB\_PERSONAL\_ACCESS\_TOKEN>"

}

}

}

}

}

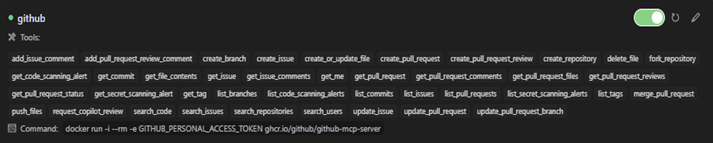
Save the file.

**Start Docker**

Ensure Docker Desktop is running. It should automatically start the container when you activate the MCP server in Cursor.

**Enable and Refresh in Cursor**

Go back to **Cursor Settings → MCP**.  
You should now see github listed as a registered MCP server.



Click **Refresh** icon if it doesn’t appear immediately.



Now, it’s time to test the Github MCP server

* 1. Press Ctrl + Alt + B to open the AI panel
  2. Press Ctrl + . and switch the mode to **Agent**
  3. Type: Connect to my GitHub repo

The GitHub MCP server should now connect and provide intelligent context from your repositories!

**Explore More**

Once connected, you can use the MCP server to:

* Search your repo history
* Summarize commits
* Answer questions using repo content

**Conclusion**

You’ve now set up an MCP server in Cursor using GitHub and Docker. This unlocks powerful AI workflows by giving LLMs structured access to your codebase.

Thanks for reading!

Today, after a long hustle I wrote my first blog. I was thinking to write the blog from very long time, but finally I took the courage and just sat down and jot down all the points. It was scary, but I feel that I had to start somewhere, maybe it’s not perfect, but everything needs to start somewhere. I don’t want to live in that regret, that I wish I could have wrote it.

The blog is about how to setup MCP servers in Cursor IDE. MCP (Model Context Protocol) is a protocol that standardizes how the applications provide context to the LLMs. It helps in connecting the AI models with different sources and tools, enhancing the power of AI agents. The invention of MCP reduced the problem of manual integration and heavy coding setup.

I want to like extend my gratitude to Medium for providing such a clean blogging platform with easier writing setup and bringing confidence in me for easy and flexible writing.

🎉 **I published my first blog today!**  
After weeks (maybe months 😅) of overthinking, self-doubt, and hesitation — I finally sat down, gathered my thoughts, and hit *publish*.

It was scary. I kept thinking:  
*“What if it’s not good enough?”*  
*“What if no one reads it?”*  
But I reminded myself: *everything has to start somewhere.* And I didn’t want to live with the regret of never even trying.

🚀 My first blog is a step-by-step guide on **how to set up MCP servers in the Cursor IDE**.  
For those unfamiliar — **MCP (Model Context Protocol)** is a powerful way to bridge AI models and tools, making LLMs smarter by giving them structured access to context. It simplifies what used to be a complex and code-heavy integration process.

📝 I'm grateful to **Medium** for offering such a clean, beginner-friendly writing experience. The platform really helped me focus on expressing my ideas — not wrestling with formatting.

This is just the beginning. I’m excited (and still a little nervous) to keep writing, keep learning, and keep sharing.

👉 If you're curious about MCP or building with AI tools, I’d love your thoughts on the post!

#FirstBlog #MCP #AI #LLMs #CursorIDE #WritingJourney #DevBlog #LearningInPublic #MediumBlog

https://medium.com/@tanmoy234am/how-to-set-up-mcp-servers-in-cursor-step-by-step-guide-17852970dbed