MCP Servers

What is MCP?

Model Context Protocol (MCP) is an open standard that allows AI assistants to connect to external tools, data sources, and services.

> *Think of it as a universal plug socket — any AI tool that supports MCP can connect to any MCP server, regardless of who built either one.*

The problem MCP solves

Without MCP, an AI assistant only knows what's in its training data or what you paste into the chat.

It can't:

- Query live infrastructure
- Read your codebase in real time
- Look up current provider documentation
- Interact with external APIs

MCP bridges that gap.

How it works

An MCP server exposes a set of **tools** that an AI agent can call — just like a human would use a CLI or browser.

```
AI Agent  →  asks a question
          →  MCP server fetches live data
          →  returns structured results to the agent
          →  agent uses that context in its response
```

The AI doesn't guess — it queries.

Why does this matter for Platform Engineering?

Platform engineers work with complex, rapidly changing systems. AI tools that rely on static training data quickly become inaccurate.

MCP servers give AI agents **live, accurate context** about:

- What providers and modules are available
- What resources exist in your infrastructure
- What the current documentation actually says

The Terraform MCP Server

HashiCorp's **Terraform MCP Server** connects AI assistants directly to the Terraform Registry.

Built by HashiCorp, it gives your AI assistant real-time access to:

- **Provider documentation** — arguments, attributes, and examples
- **Module search** — find and evaluate community modules
- **Resource details** — exact schema for any resource type
- **Policy libraries** — search Sentinel policies

https://github.com/hashicorp/terraform-mcp-server

Without it vs with it

Without the Terraform MCP Server

> *"Create an Azure Container App" — the AI guesses at the resource schema based on training data, which may be outdated or incomplete. And may not follow the best up to date information when provisioning infrastructre*

With the Terraform MCP Server

> *The AI queries the live registry, retrieves the exact current schema for `azurerm_container_app`, and generates accurate, up-to-date Terraform.*

Why use it?

- **Accuracy** — resources are generated from live registry docs, not stale training data
- **Speed** — no switching between browser, docs, and editor
- **Confidence** — the output maps to what the provider actually supports today
- **Learning** — as you work, the AI explains what each argument does and why

For a Platform Engineer writing Terraform every day, this is a significant productivity gain.

How to add it to VS Code

Add to your VS Code User Settings (JSON):

Create a foler/file in your repo called `.vscode/mcp.json`

Add the below content and save, you should see a `Start` button appear.

```
{
  "mcp": {
    "servers": {
      "terraform": {
        "command": "docker",
        "args": [
          "run", "-i", "--rm",
          "hashicorp/terraform-mcp-server"
        ]
      }
    }
  }
}
```

MCP servers are growing fast

The Terraform MCP Server is one of hundreds now available — covering AWS, Kubernetes, databases, monitoring tools, and more.

As a Platform Engineer, understanding how to **configure and use MCP servers** is increasingly part of the toolkit — it's how you get the most out of AI assistants in a real engineering environment.