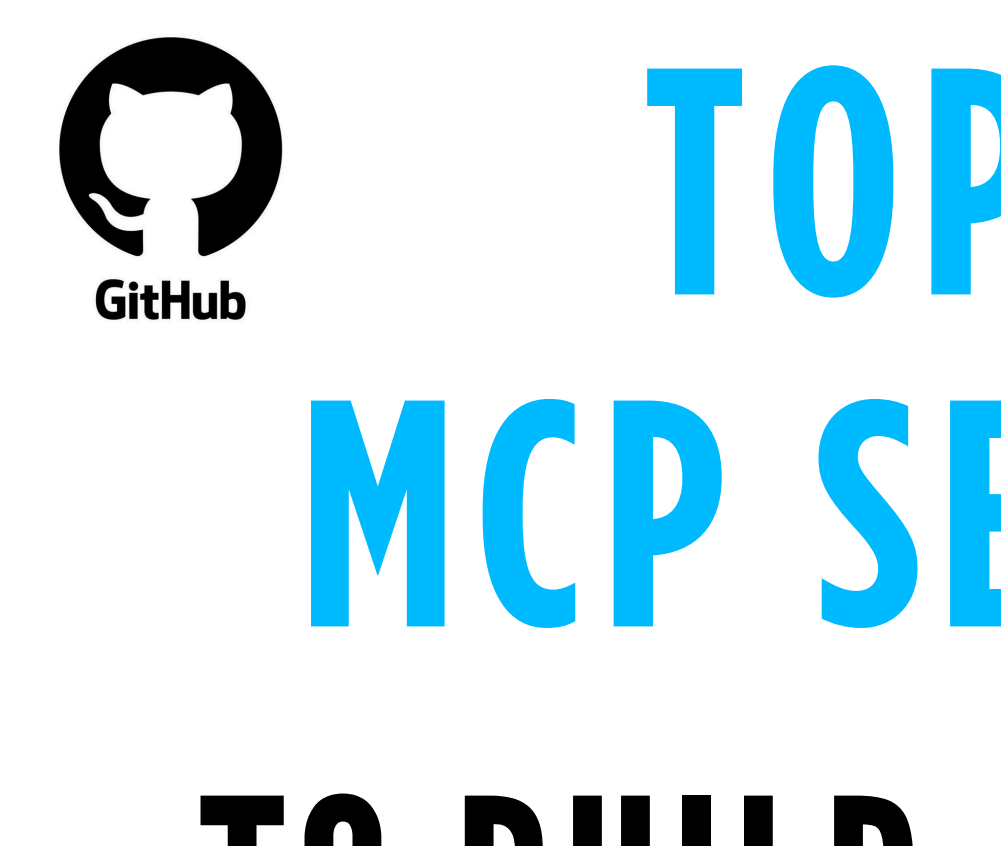
GitHub

TOP 20 MCP SERVERS

TO BUILD AI AGENTS



SLIDE TO EXPLORE

linkedin.com/in/that-aum

Table of Contents

1. Sequential Thinking MCP
2. GitHub MCP
3. Serena MCP
4. DesktopCommander MCP
5. File System MCP
6. Docker MCP
7. Supabase MCP
8. Puppeteer MCP
9. Playwright MCP
10. Firecrawl MCP
11. DuckDuckGo MCP
12. Memory Bank MCP
13. Knowledge Graph Memory MCP
14. Markdownify MCP
15. Graphiti MCP
16. Perplexity MCP
17. Magic UI MCP
18. Zen MCP
19. Figma MCP
20. MCP Compass

1

Sequential Thinking MCP Server

An MCP server implementation that provides a tool for dynamic and reflective problem-solving through a structured thinking process.

Features

- Break down complex problems into manageable steps
- Revise and refine thoughts as understanding deepens
- Branch into alternative paths of reasoning
- Adjust the total number of thoughts dynamically
- Generate and verify solution hypotheses

Tool

sequential_thinking

Facilitates a detailed, step-by-step thinking process for problem-solving and analysis.

Inputs:

- `thought` (string): The current thinking step
- `nextThoughtNeeded` (boolean): Whether another thought step is needed
- `thoughtNumber` (integer): Current thought number
- `totalThoughts` (integer): Estimated total thoughts needed
- `isRevision` (boolean, optional): Whether this revises previous thinking
- `revisesThought` (integer, optional): Which thought is being reconsidered
- `branchFromThought` (integer, optional): Branching point thought number
- `branchId` (string, optional): Branch identifier
- `needsMoreThoughts` (boolean, optional): If more thoughts are needed

Usage

The Sequential Thinking tool is designed for:

- Breaking down complex problems into steps
- Planning and design with room for revision
- Analysis that might need course correction
- Problems where the full scope might not be clear initially
- Tasks that need to maintain context over multiple steps
- Situations where irrelevant information needs to be filtered out

2 GitHub MCP Server

The GitHub MCP Server is a Model Context Protocol (MCP) server that provides seamless integration with GitHub APIs, enabling advanced automation and interaction capabilities for developers and tools.

Use Cases

- Automating GitHub workflows and processes.
- Extracting and analyzing data from GitHub repositories.
- Building AI powered tools and applications that interact with GitHub's ecosystem.

Remote GitHub MCP Server

The remote GitHub MCP Server is hosted by GitHub and provides the easiest method for getting up and running. If your MCP host does not support remote MCP servers, don't worry! You can use the local version of the GitHub MCP Server instead.

Prerequisites

1. An MCP host that supports the latest MCP specification and remote servers, such as VS Code.

Installation

Usage with VS Code

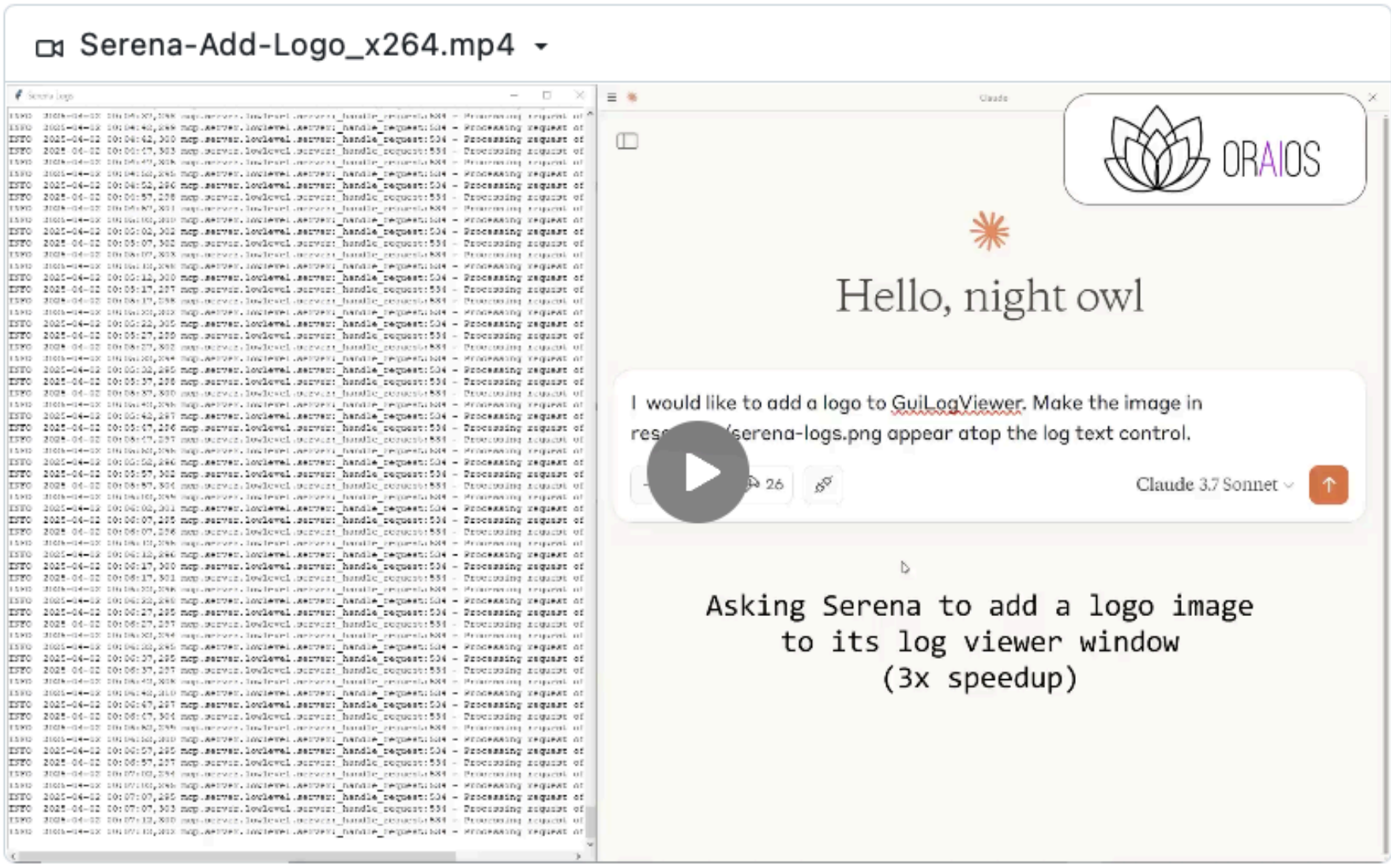
For quick installation, use one of the one-click install buttons above. Once you complete that flow, toggle Agent mode (located by the Copilot Chat text input) and the server will start. Make sure you're using VS Code 1.101 or later for remote MCP and OAuth support.

3 Serena MCP

- 🚀 Serena is a powerful **coding agent toolkit** capable of turning an LLM into a fully-featured agent that works **directly on your codebase**.
- 🔧 Serena provides essential **semantic code retrieval and editing tools** that are akin to an IDE's capabilities, extracting code entities at the symbol level and exploiting relational structure.
- 🆓 Serena is **free & open-source**, enhancing the capabilities of LLMs you already have access to free of charge.

Demonstration

Here is a demonstration of Serena implementing a small feature for itself (a better log GUI) with Claude Desktop. Note how Serena's tools enable Claude to find and edit the right symbols.



Serena is under active development! See the latest updates, upcoming features, and lessons learned to stay up to date.

Updates Roadmap Lessons Learned

4 DesktopCommander MCP

Search, update, manage files and run terminal commands with AI

Work with code and text, run processes, and automate tasks, going far beyond other AI editors - without API token costs.

Desktop Commander MCP

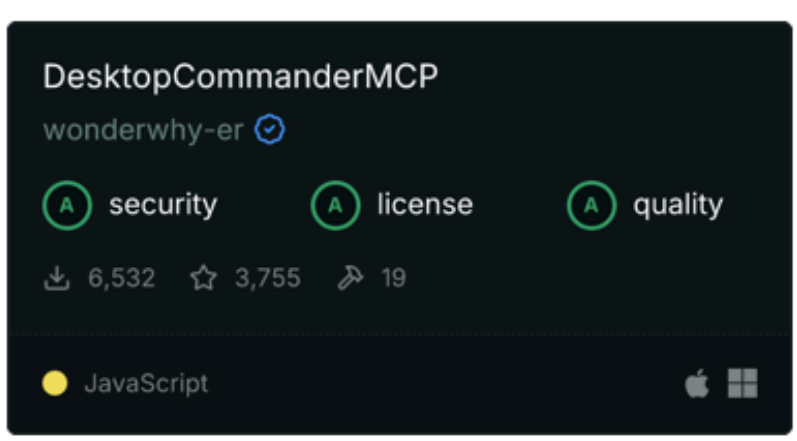


Table of Contents

- Features
- Installation
- Usage
- Handling Long-Running Commands
- Work in Progress and TODOs
- Sponsors and Supporters
- Website
- Media
- Testimonials
- Frequently Asked Questions
- Contributing
- License

All of your AI development tools in one place. Desktop Commander puts all dev tools in one chat. Execute long-running terminal commands on your computer and manage processes through Model Context Protocol (MCP). Built on top of MCP Filesystem Server to provide additional search and replace file editing capabilities.

5 FileSystem MCP

Node.js server implementing Model Context Protocol (MCP) for filesystem operations.

Features

- Read/write files
- Create/list/delete directories
- Move files/directories
- Search files
- Get file metadata
- Dynamic directory access control via Roots

Directory Access Control

The server uses a flexible directory access control system. Directories can be specified via command-line arguments or dynamically via Roots.

Method 1: Command-line Arguments

Specify Allowed directories when starting the server:

```
mcp-server-filesystem /path/to/dir1 /path/to/dir2
```

Method 2: MCP Roots (Recommended)

MCP clients that support Roots can dynamically update the Allowed directories.

Roots notified by Client to Server, completely replace any server-side Allowed directories when provided.

Important: If server starts without command-line arguments AND client doesn't support roots protocol (or provides empty roots), the server will throw an error during initialization.

This is the recommended method, as this enables runtime directory updates via `roots/list_changed` notifications without server restart, providing a more flexible and modern integration experience.

6 Docker MCP

Jean-Laurent de Morlhon

Yiwen Xu

Model Context Protocols (MCPs) are quickly becoming the standard for connecting AI agents to external tools, but the developer experience hasn't caught up. Discovery is fragmented, setup is clunky, and security is too often bolted on last. Fixing this experience isn't a solo mission—it will take an industry-wide effort. A secure, scalable, and trusted MCP ecosystem demands collaboration across platforms and vendors.

That's why we're excited to announce Docker MCP Catalog and Toolkit are now available in Beta. The Docker MCP Catalog, now a part of Docker Hub, is your starting point for discovery, surfacing a curated set of popular, containerized MCP servers to jumpstart agentic AI development. But discovery alone isn't enough. That's where the MCP Toolkit comes in. It simplifies installation, manages credentials, enforces access control, and secures the runtime environment. Together, Docker MCP Catalog and MCP Toolkit give developers and teams a complete foundation for working with MCP tools, making them easier to find, safer to use, and ready to scale across projects and teams.

We're partnering with some of the most trusted names in cloud, developer tooling, and AI, including Stripe, Elastic, Heroku, Pulumi, Grafana Labs, Kong Inc., Neo4j, New Relic, Continue.dev, and many more, to shape a secure ecosystem for MCP tools. With a one-click connection right from Docker Desktop to leading MCP clients like Gordon (Docker AI Agent), Claude, Cursor, VSCode, Windsurf, continue.dev, and Goose, building powerful, intelligent AI agents has never been easier.

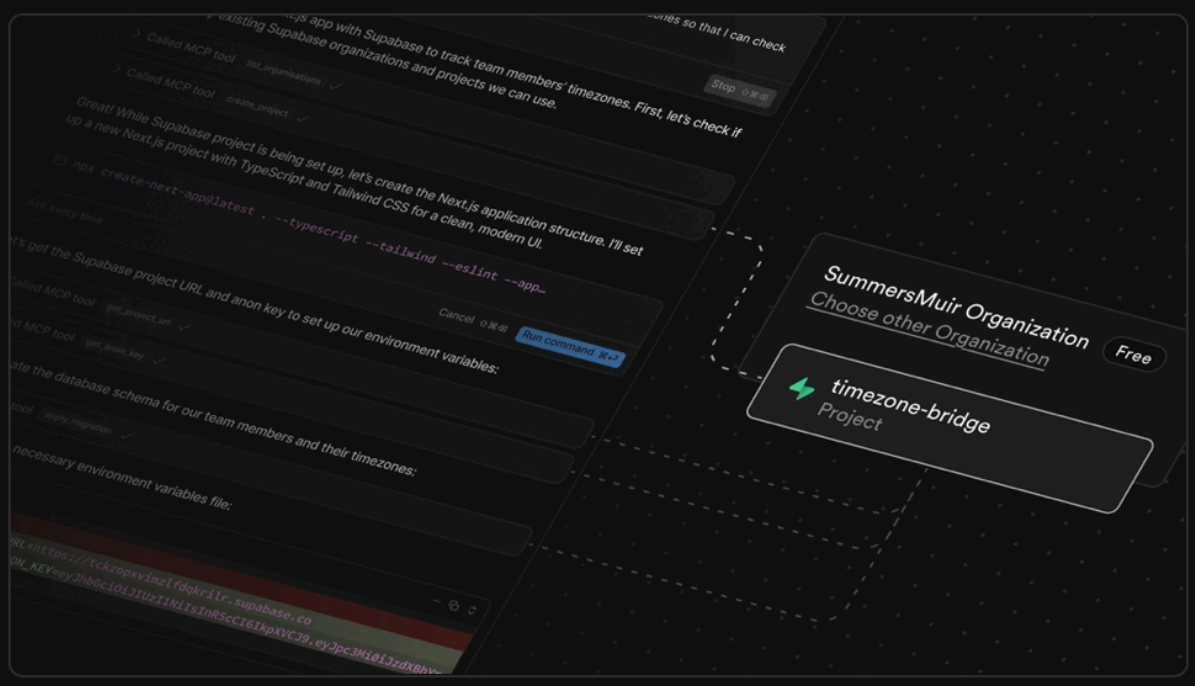
This aligns perfectly with our mission. Docker pioneered the container revolution, transforming how developers build and deploy software. Today, over 20 million registered developers rely on Docker to build, share, and run modern applications. Now, we're bringing that same trusted experience to the next frontier: Agentic AI with MCP tools.

7 Supabase MCP

Supabase MCP Server

04 Apr 2025 • 9 minute read

Greg Richardson
Engineering



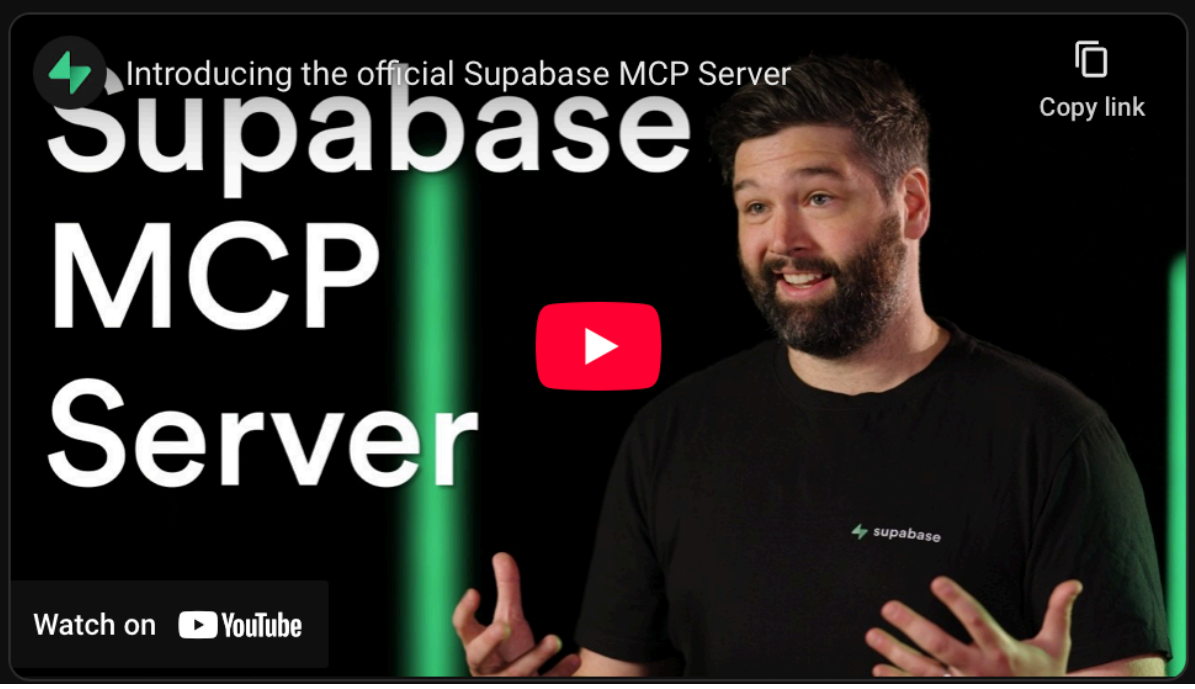
launch-week

On this page

What is an MCP Server?
Tools
Setup
How does MCP work?
The MCP standard
Resources and prompts
What's next?
Create and deploy Edge Functions
Native authorization
Better schema discovery
More protections
Get started with Supabase MCP

Share this article

We are launching an official Supabase MCP server. You can use this server to connect your favorite AI tools (such as Cursor or Claude) directly with Supabase.



8 Puppeteer MCP

A Model Context Protocol server that provides browser automation capabilities using Puppeteer. This server enables LLMs to interact with web pages, take screenshots, and execute JavaScript in a real browser environment.

> **Caution**
>
> This server can access local files and local/internal IP addresses since it runs a browser on your machine. Exercise caution when using this MCP server to ensure this does not expose any sensitive data.

Components

Tools

- **puppeteer_navigate**
  - Navigate to any URL in the browser
  - Inputs:
    - `url` (string, required): URL to navigate to
    - `launchOptions` (object, optional): PuppeteerJS LaunchOptions. Default null. If changed and not null, browser restarts. Example: `{ headless: true, args: ['--user-data-dir="C:/Data"'] }`
    - `allowDangerous` (boolean, optional): Allow dangerous LaunchOptions that reduce security. When false, dangerous args like `--no-sandbox`, `--disable-web-security` will throw errors. Default false.
- **puppeteer_screenshot**
  - Capture screenshots of the entire page or specific elements
  - Inputs:
    - `name` (string, required): Name for the screenshot
    - `selector` (string, optional): CSS selector for element to screenshot
    - `width` (number, optional, default: 800): Screenshot width
    - `height` (number, optional, default: 600): Screenshot height
    - `encoded` (boolean, optional): If true, capture the screenshot as a base64-encoded data URI (as text) instead of binary image content. Default false.
- **puppeteer_click**
  - Click elements on the page
  - Input: `selector` (string): CSS selector for element to click

9 Playwright MCP

A Model Context Protocol (MCP) server that provides browser automation capabilities using Playwright. This server enables LLMs to interact with web pages through structured accessibility snapshots, bypassing the need for screenshots or visually-tuned models.

Key Features

- **Fast and lightweight**. Uses Playwright's accessibility tree, not pixel-based input.
- **LLM-friendly**. No vision models needed, operates purely on structured data.
- **Deterministic tool application**. Avoids ambiguity common with screenshot-based approaches.

Requirements

- Node.js 18 or newer
- VS Code, Cursor, Windsurf, Claude Desktop, Goose or any other MCP client

Getting started

First, install the Playwright MCP server with your client. A typical configuration looks like this:

```
{
  "mcpServers": {
    "playwright": {
      "command": "npx",
      "args": [
        "@playwright/mcp@latest"
      ]
    }
  }
}
```

Install Server VS Code Install Server VS Code Insiders

10 Firecrawl MCP

A Model Context Protocol (MCP) server implementation that integrates with Firecrawl for web scraping capabilities.

> Big thanks to @vrknetha, @knacklabs for the initial implementation!

Features

- Web scraping, crawling, and discovery
- Search and content extraction
- Deep research and batch scraping
- Automatic retries and rate limiting
- Cloud and self-hosted support
- SSE support

> Play around with our MCP Server on MCP.so's playground or on Klavis AI.

Installation

Running with npx

```
env FIRECRAWL_API_KEY=fc-YOUR_API_KEY npx -y firecrawl-mcp
```

Manual Installation

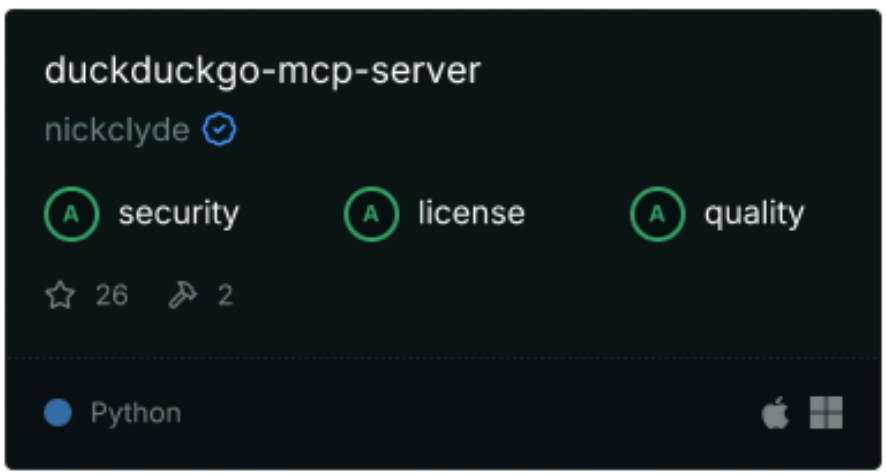
```
npm install -g firecrawl-mcp
```

Running on Cursor

Configuring Cursor 🖥 Note: Requires Cursor version 0.45.6+ For the most up-to-date configuration instructions, please refer to the official Cursor documentation on configuring MCP servers: Cursor MCP Server Configuration Guide

11 DuckDuckGo Search MCP

A Model Context Protocol (MCP) server that provides web search capabilities through DuckDuckGo, with additional features for content fetching and parsing.



Features

- **Web Search**: Search DuckDuckGo with advanced rate limiting and result formatting
- **Content Fetching**: Retrieve and parse webpage content with intelligent text extraction
- **Rate Limiting**: Built-in protection against rate limits for both search and content fetching
- **Error Handling**: Comprehensive error handling and logging
- **LLM-Friendly Output**: Results formatted specifically for large language model consumption

Installation

Installing via Smithery

To install DuckDuckGo Search Server for Claude Desktop automatically via Smithery:

```
npx -y @smithery/cli install @nickclyde/duckduckgo-mcp-server --cli
```

12

Memory Bank MCP Server

A Model Context Protocol (MCP) server implementation for remote memory bank management, inspired by Cline Memory Bank.

Overview

The Memory Bank MCP Server transforms traditional file-based memory banks into a centralized service that:

- Provides remote access to memory bank files via MCP protocol
- Enables multi-project memory bank management
- Maintains consistent file structure and validation
- Ensures proper isolation between project memory banks

Features

- **Multi-Project Support**
  - Project-specific directories
  - File structure enforcement
  - Path traversal prevention
  - Project listing capabilities
  - File listing per project
- **Remote Accessibility**
  - Full MCP protocol implementation
  - Type-safe operations
  - Proper error handling
  - Security through project isolation
- **Core Operations**
  - Read/write/update memory bank files
  - List available projects
  - List files within projects
  - Project existence validation
  - Safe read-only operations

13 Knowledge Graph Memory Server

A basic implementation of persistent memory using a local knowledge graph. This lets Claude remember information about the user across chats.

Core Concepts

Entities

Entities are the primary nodes in the knowledge graph. Each entity has:

- A unique name (identifier)
- An entity type (e.g., "person", "organization", "event")
- A list of observations

Example:

```
{
  "name": "John_Smith",
  "entityType": "person",
  "observations": ["Speaks fluent Spanish"]
}
```

Relations

Relations define directed connections between entities. They are always stored in active voice and describe how entities interact or relate to each other.

Example:

```
{
  "from": "John_Smith",
  "to": "Anthropic",
  "relationType": "works_at"
}
```

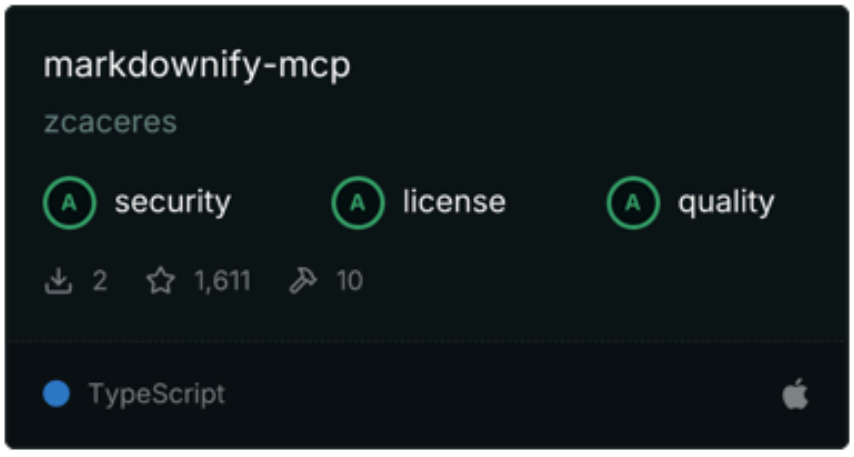
Observations

Observations are discrete pieces of information about an entity. They are:

- Stored as strings
- Attached to specific entities
- Can be added or removed independently
- Should be atomic (one fact per observation)

14 Markdownify MCP Server

Markdownify is a Model Context Protocol (MCP) server that converts various file types and web content to Markdown format. It provides a set of tools to transform PDFs, images, audio files, web pages, and more into easily readable and shareable Markdown text.



Features

- Convert multiple file types to Markdown:
  - PDF
  - Images
  - Audio (with transcription)
  - DOCX
  - XLSX
  - PPTX
- Convert web content to Markdown:
  - YouTube video transcripts
  - Bing search results
  - General web pages
- Retrieve existing Markdown files

15

Graphiti MCP Server

*Fork & extension of the official [`getzep/graphiti` ********** MCP server]—adding multi-server, single-DB support and a DX-focused CLI.*

> **Build per-project temporal knowledge graphs that your AI agents can query over the [Model Context Protocol]—all in one command.**

Why this repo exists

Graphiti already turns unstructured text into a **temporal graph** stored in Neo4j—each server ingests text, extracts entities & relationships via LLMs, and records every change as time-stamped episodes so agents can ask versioned questions, but most IDEs and agent frameworks (Cursor, VS Code, LangGraph, Autogen, …) speak **MCP**—they expect an HTTP/SSE endpoint that they can list in a `mcp.json` file.
Typical workflows force you to hand-roll a dedicated server for every project. To remove that manual step, this CLI auto-generates a *Docker Compose* file that spins up:

- **one Neo4j instance** (shared storage)
- **one "root" MCP server** (playground / smoke tests)
- **N project-scoped MCP servers**—each with its own `group_id` , entity rules and OpenAI model

Unlike the upstream example, which assumes **one server per docker-compose file**, this fork automates **N servers against a single Neo4j** so you get:

Benefit	Why it matters
Project isolation	Different extraction rules or models can't collide.
Editor auto-discovery	`.cursor/mcp.json` is rewritten with the right port for each project —open the repo, tools appear.
Crash containment	A runaway prompt that floods the graph only takes down *its* container.
Zero-downtime tweaks	Hot-swap entity YAML or LLM model for *project B* without restarting *project A*.

If your workload is small and homogeneous you *can* run a single server—just comment out the project entries in `mcp-projects.yaml` . The defaults aim for safety and DX first.

16

Preplexity MCP

Guides

Integrating MCP with Perplexity's Sonar API

Learn about the MCP server implementation for the Sonar API.

What is MCP?

The **Model Context Protocol (MCP)** is an open standard designed to connect AI assistants with the systems where data lives. By providing a universal interface for AI applications to access structured and unstructured data, MCP eliminates the need for custom integrations, enabling AI models to retrieve real-time, relevant information more efficiently. Learn more about MCP here.

MCP Server for Perplexity's Sonar API

Our **MCP server implementation** for Sonar allows any AI-powered tool to perform real-time, web-wide research using Perplexity's powerful search engine. This server acts as a bridge between AI applications and the Sonar, enabling seamless integration for retrieving and synthesizing relevant, up-to-date information from the web.

How It Works

- The **Perplexity Ask MCP Server** follows MCP's **open standard**, allowing any AI assistant or automation tool to connect to the **Sonar API** for live web searches.
- AI models can query the server for information retrieval, leveraging Perplexity's search capabilities to return the most relevant insights.

Example: Using MCP with Claude

Claude is one example of how this MCP server can be used effectively. When connected to Claude, the **Perplexity Ask MCP Server** enables it to **perform live web searches** and return **highly relevant, up-to-date responses**, enhancing its ability to provide real-time knowledge.

17 Magic UI MCP

Magic UI now has an official MCP server 🎉.

MCP is an open protocol that standardizes how applications provide context to LLMs.

This is useful for Magic UI because you can now give your AI-assisted IDE direct access to all Magic UI components so that it can generate code with minimal errors.

CLI Manual

1 **Installation**

Cursor Windsurf Claude Cline Roo-Cline

pnpm npm yarn bun

```
pnpm dlx @magicuidesign/cli@latest install cursor
```

2 **Restart your IDE**

Usage

You can now ask your IDE to use any Magic UI component. Here are some examples:

- "Add a blur fade text animation"
- "Add a grid background"
- "Add a vertical marquee of logos"

18 Zen MCP

The ultimate development partners for your favorite Coding Agent (Claude OR Gemini CLI) - a Model Context Protocol server that gives you access to multiple AI models for enhanced code analysis, problem-solving, and collaborative development.

Features true AI orchestration with conversations that continue across workflows - Give Claude a complex *workflow* and let it orchestrate between models automatically. Claude stays in control, performs the actual work, but gets perspectives from the best AI for each subtask. With tools like `planner` for breaking down complex projects, `analyze` for understanding codebases, `codereview` for audits, `refactor` for improving code structure, `debug` for solving complex problems, and `precommit` for validating changes, Claude can switch between different tools *and* models mid-conversation, with context carrying forward seamlessly.

Example Workflow - Claude Code:

1. `Perform a codereview using gemini pro and o3 and use planner to generate a detailed plan, implement the fixes and do a final precommit check by continuing from the previous codereview`
2. This triggers a `codereview` workflow where Claude walks the code, looking for all kinds of issues
3. After multiple passes, collects relevant code and makes note of issues along the way
4. Maintains a `confidence` level between `exploring`, `low`, `medium`, `high` and `certain` to track how confidently it's been able to find and identify issues
5. Generates a detailed list of critical -> low issues
6. Shares the relevant files, findings, etc with **Gemini Pro** to perform a deep dive for a second `codereview`
7. Comes back with a response and next does the same with o3, adding to the prompt if a new discovery comes to light
8. When done, Claude takes in all the feedback and combines a single list of all critical -> low issues, including good patterns in your code. The final list includes new findings or revisions in case Claude misunderstood or missed something crucial and one of the other models pointed this out
9. It then uses the `planner` workflow to break the work down into simpler steps if a major refactor is required
10. Claude then performs the actual work of fixing highlighted issues
11. When done, Claude returns to Gemini Pro for a `precommit` review

19 Figma MCP

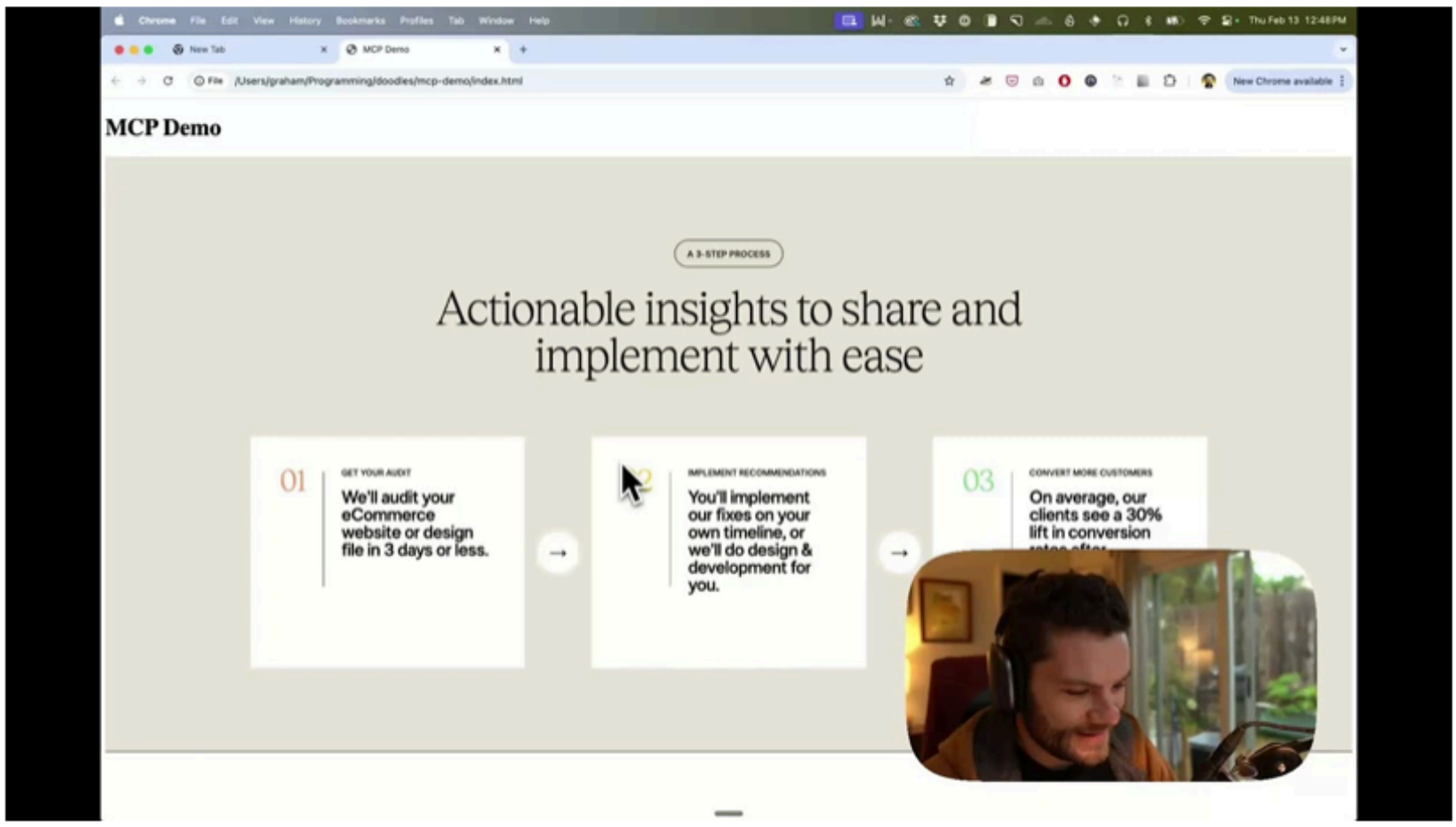
Give Cursor and other AI-powered coding tools access to your Figma files with this Model Context Protocol server.

When Cursor has access to Figma design data, it's **way** better at one-shotting designs accurately than alternative approaches like pasting screenshots.

See quickstart instructions →

Demo

Watch a demo of building a UI in Cursor with Figma design data



How it works

1. Open your IDE's chat (e.g. agent mode in Cursor).
2. Paste a link to a Figma file, frame, or group.
3. Ask Cursor to do something with the Figma file—e.g. implement the design.
4. Cursor will fetch the relevant metadata from Figma and use it to write your code.

This MCP server is specifically designed for use with Cursor. Before responding with context from the Figma API, it simplifies and translates the response so only the most relevant layout and styling information is provided to the model.

20 MCP Compass

Experience MCP Compass 🌐

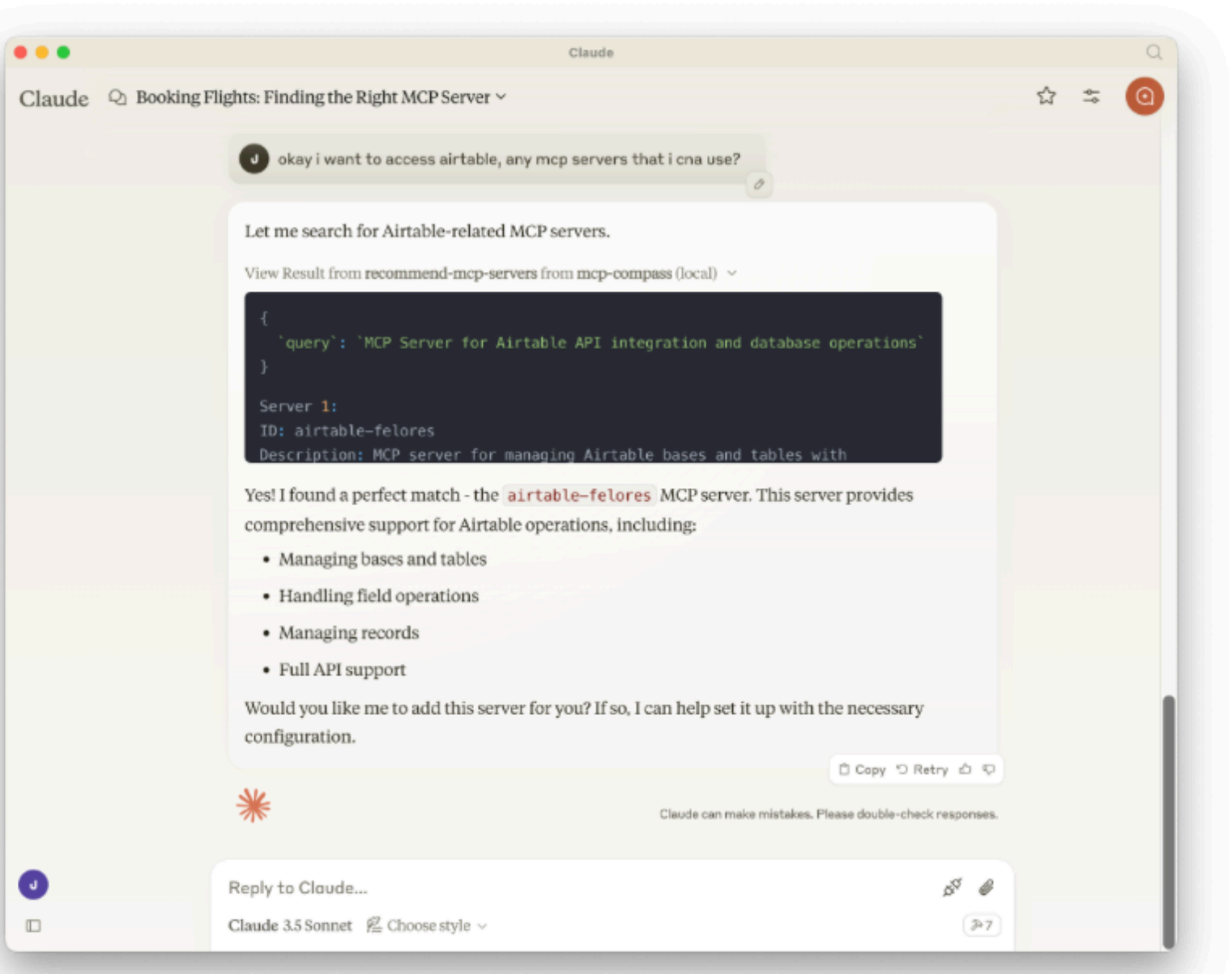
You can now experience MCP discovery directly on our website!
👉 Explore MCP Compass 👈

Dive in to discover the power of MCP services in action with real-time recommendations and insights.

What is this? 🤔

MCP Compass is a discovery & recommendation service that helps you explore Model Context Protocol servers. It acts as a smart guide that helps AI assistants find and understand available MCP services out there based on **natural language queries**, making it easier to discover and utilize the right tools for specific tasks.

Quick Example



Interested in
more content like this?

Follow me :

OM NALINDE

linkedin.com/in/that-aum