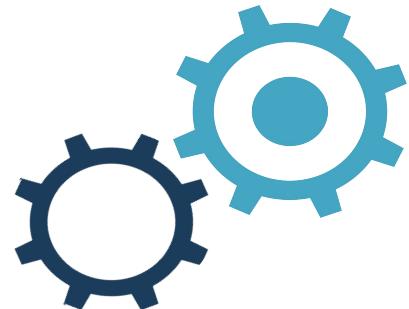


Turn Your Web Server Into An MCP Server



Agenda

1. About this workshop
2. Refresher on Web Servers and APIs
3. Overview of MCPs Clients and Servers
4. The Different Types of MCP servers
5. MCP Server Design
6. MCP In The Real World
7. Creating an MCP Server
8. Adapting a REST API to Create an MCP Server
9. Get an MCP Client to Use an MCP Server
10. Upcoming MCP Features & Developments

About this workshop

- Enhance existing back-end web dev skills
- Add a new trendy buzzword to your portfolio
- Learn another way that AI can be integrated into products
- Ensure that you're still useful to companies chasing trends
even if you're anti-AI

Refresher on Web Servers and APIs

- Servers are computers that run apps that others can interact with
- APIs are interfaces to other applications
- Web servers are apps that let clients (people or apps) send in requests and receive responses based on those requests
- Web API servers typically exist to let clients interact with complex systems through web requests

Refresher on Web Servers and APIs



Refresher on Web Servers and APIs

```
1 // server.mjs
2 import { createServer } from 'node:http';
3
4 const server = createServer((req, res) => {
5   res.writeHead(200, { 'Content-Type': 'text/plain' });
6   res.end('Hello World!\n');
7 });
8
9 // starts a simple http server locally on port 3000
10 server.listen(3000, '127.0.0.1', () => {
11   console.log('Listening on 127.0.0.1:3000');
12 });
13
14 // run with 'node server.mjs'
```

NodeJS
web server
nodejs.org

Refresher on Web Servers and APIs

```
const express = require('express')
const app = express()
const port = 3000

app.get('/', (req, res) => {
  res.send('Hello World!')
})

app.listen(port, () => {
  console.log(`Example app listening on port ${port}`)
})
```

ExpressJS web
server
expressjs.com

Refresher on Web Servers and APIs

```
src > JS index.js > ...
1  const express = require("express");
2  const app = express();
3
4  require("dotenv").config();  6.8k (gzipped: 2.9k)
5  // dotenv could be removed if we run the app with:
6  // node --env-file=.env src/index.js
7  const PORT = process.env.PORT || 3456;
8
9  app.get("/", (request, response) => {
10    response.json({
11      message:"Hello, world!"
12    });
13  });
14
15 app.listen(PORT, () => {
16   console.log("Server running on port:" + PORT);
17 });
18
```

ExpressJS
web API
server

Refresher on Web Servers and APIs

- HTTP verbs
 - e.g. GET, POST, PATCH, DELETE
- Resource-focused design
 - e.g. CRUD operations
- Allows interaction with a variety of clients

Overview of MCPs Clients and Servers

- Model Context Protocol (MCP) is a protocol, not an AI or LLM or ML!
- MCP clients = LLM apps, like ChatGPT's web app
- MCP servers = literally any server that implements MCP
- **The purpose of MCP is to give LLMs actual functionality!**

Overview of MCPs Clients and Servers

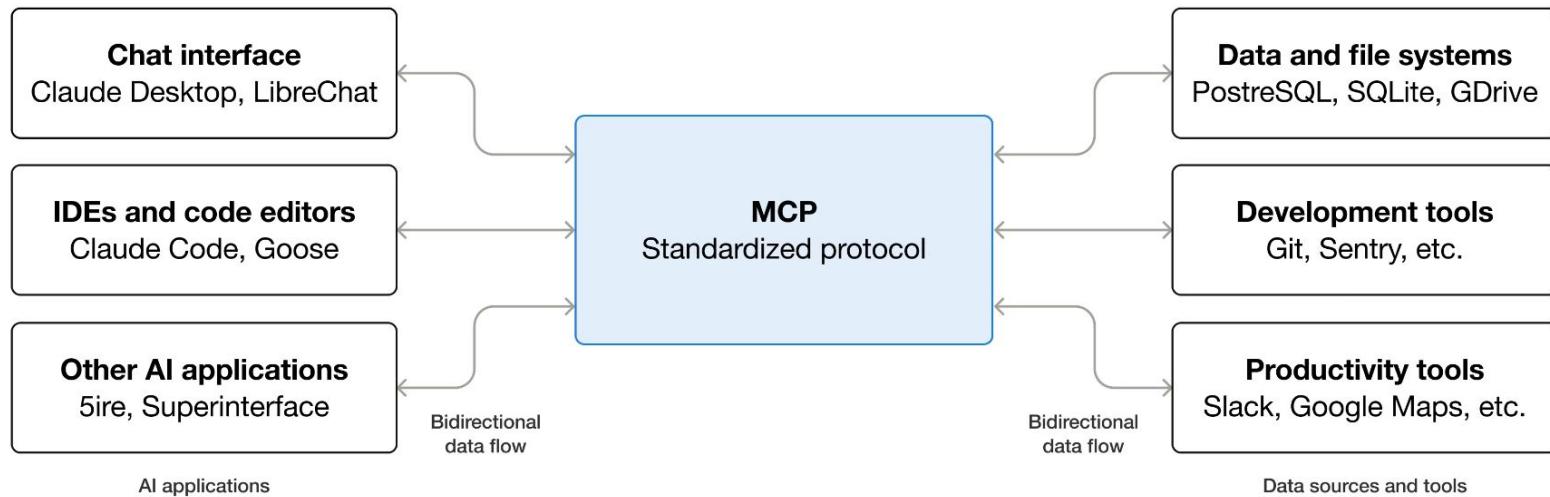
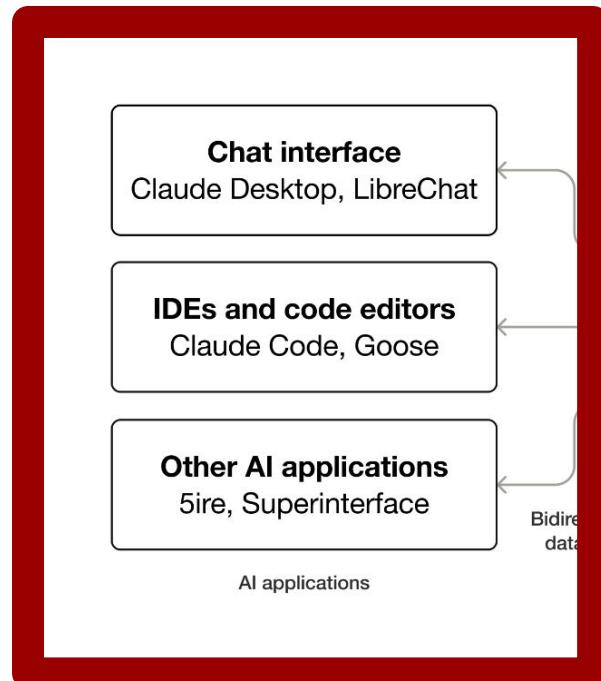


Image from modelcontextprotocol.io/docs/getting-started/intro

Overview of MCPs Clients and Servers

AI/LLM hype stuff



Perfectly regular software development

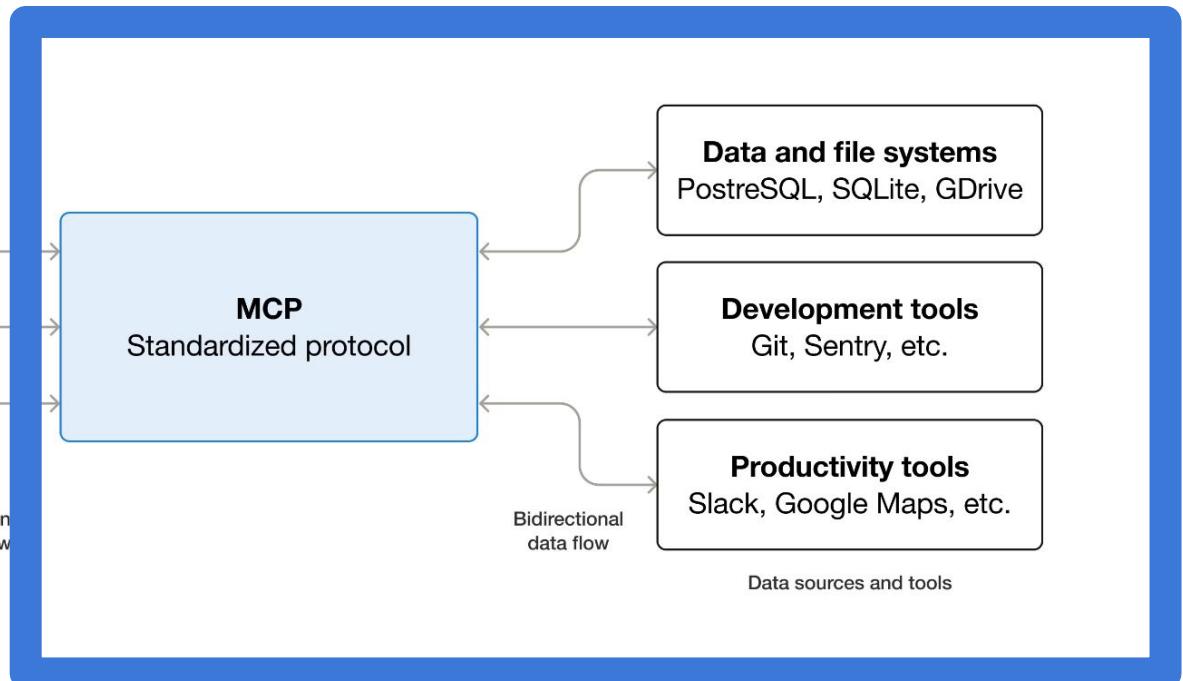


Image from modelcontextprotocol.io/docs/getting-started/intro

The Different Types of MCP servers

REST APIs Running As MCP Servers

- Nothing difficult to this if you adhere to D.R.Y coding principles
- Does everything that a REST API does, but wrapped in MCP

Built-For-MCP MCP Servers

- Actual thought about what the MCP server exposes for usage
- Organises functionality by action, not by resource

MCP Server Design

- REST APIs are commonly granular, especially for microservices!
- MCP clients have error rates, varying per LLM model and client
- MCP client actions that require multiple “tool calls” have that error rate applied for each tool call

Therefore...

- MCP servers should **not** be granular like a REST API!

MCP Server Design

Web App

User submits a search form to search for all articles with a specific tag

REST API

GET /api/articles/?searchTerm=coding

GET /api/users/me

POST /api/auth/sessionExtender

Functionality

Get filtered article array

Get user profile picture and name for client display

Ensure the user has a fresh JWT for their session

MCP Server Design

MCP Client

User writes
that they
want to
search for all
articles with
a specific tag

MCP Server

Run tool named
“search-articles”

Functionality

Get filtered
article array

Get user profile
picture and
name for client
display

Ensure the user
has a fresh JWT
for their session

MCP Server Design

- Fewer tool calls = better LLM tool selection
- Fewer tool calls = more-accurate responses in complex requests
- Fewer tool calls = fewer tokens used = better pricing of LLM services

This is for a pretty simple server [a calculator API], but let's use this as a ballpark.

A single action requiring 5 tool calls using these estimates would cost something like 23,640 input tokens (vs. 2,927 for a single tool call) and 600 output tokens (vs. 200 for a single tool call).

Using [current Claude Sonnet 4 pricing](#), this comes out to \$0.07 in input tokens per action (vs. ~\$0.00878) and \$0.009 in output tokens per action vs. \$0.003.

In total each action with five simple tool calls costs \$0.08 versus ~\$0.012 per action, **an almost 7x increase**. These numbers seem small, but this is just one action per user.

<https://thesignalpath.xyz/stop-generating-mcp-servers-from-rest-apis/>

MCP Server Design

- Tools:
 - The functionality of the server, can be anything and everything
- Resources:
 - Read-only data that servers can serve up to clients
- Prompts:
 - Templates or hints for MCP clients when using the server

MCP Server Design

- Tools exposed instead of API endpoints
- Action-focused design
 - e.g. log in to XYZ News and show me the latest 10 articles in ABC category
- Allows interaction with a variety of clients

MCP In The Real World - PayPal

 [create invoice with PayPal](#)

I'd be happy to help you create an invoice using PayPal. To proceed, I'll need information:

1. Your email address (invoicer's email)
2. The recipient's email address
3. The product or service name
4. The cost of the product/service
5. Tax percentage (if applicable)
6. Discount percentage (if applicable)

Once you provide these details, I'll create the invoice for you through PayPal



Claude can make mistakes. Please double-check responses.

[Download PDF](#)

Invoice from Service Provider

Service Provider [REDACTED]@gmail.com

Invoice #1592467803
Issued: Apr 1, 2025
Due: Apr 11, 2025

Amount due: \$262.50

Select payment method:

\$262.50 

PayPal 
venmo 
Pay Later 
Debit or Credit Card 

Powered by **PayPal**

You understand that your data will be subject to the PayPal [Privacy Statement](#).

Items

House Cleaning	\$250.00
1 x \$250.00 Sales Tax (10%) (\$25.00) Complete house cleaning service	

Seller note to customer
Thank you for your business.

Don't recognize this invoice?
 [Report this invoice](#)

If you don't recognize this invoice, report it. PayPal would never use an invoice or a money request to ask you for your account credentials.

paypal.ai/docs/tools/mcp-quickstart

MCP In The Real World - PayPal

PayPal Docs

Get Started Accept Payments

Use Tools and AI

Introduction

Quickstart guides

Agent toolkit

MCP server

LLM integration

Reference

Agent tools

Best practices

Agents for commerce

Prompts

Catalog management

Dispute management

Invoices

Payments

Reporting and insights

Shipment tracking

Subscription management

Additional tools for remote MCP
server

Invoices

Functionality	Description	Parameters	Sample Prompt
<code>create_invoice</code>	Create a new invoice in the PayPal system.	<code>recipient_email</code> : Email address of the invoice recipient <code>items</code> : List of items or services to include in the invoice: <ul style="list-style-type: none">- <code>name</code> (string): Name of the item- <code>quantity</code> (number): Quantity of the item- <code>unit_price</code> (number): Price per unit of the item	Create an invoice for <code>{customer_email}</code> including 2 hours of consulting at \$150 per hour.

paypal.ai/docs/tools/agent-tools-ref

MCP In The Real World - ANZ Bank



Amazon Web Services (AWS)

10,304,980 followers

3mo

...

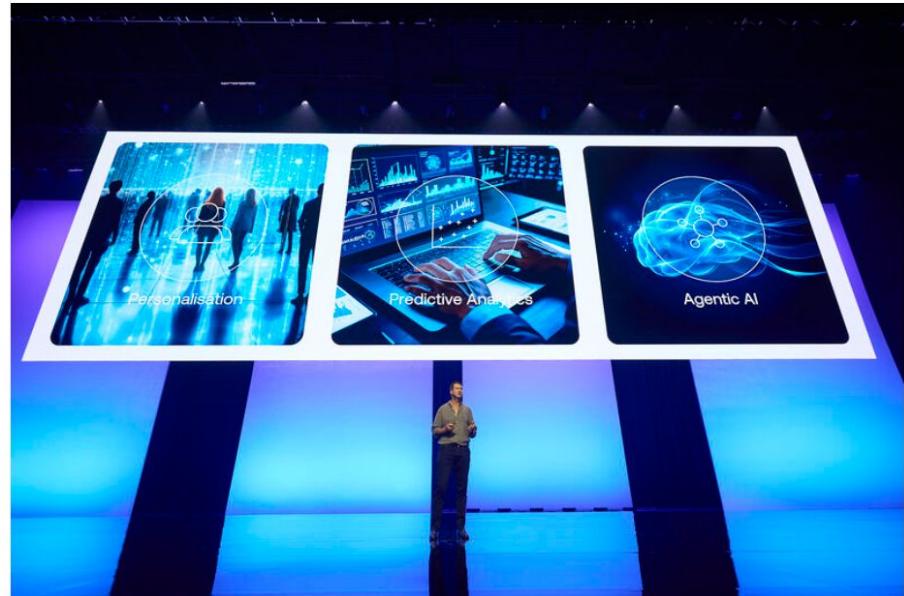
Transforming Financial Services with AI at #AWSSummit Sydney.💡

Fascinating presentation by **ANZ** Bank showcasing AMIE - their cutting-edge AI sales assistant powered by Amazon Bedrock and Anthropic Claude Sonnet 3.5.

Breakthrough features:

- ◆ Conversational FX and Deposits Data analysis
- ◆ Real-time market insights
- ◆ Seamless research integration
- ◆ Enhanced customer experience

This isn't just about banking - it's about reimagining how AI can transform customer interactions across industries.



MCP In The Real World - Commonwealth Bank

Commonwealth Bank replaces dozens of call centre jobs with AI chatbot

By Audrey Court

Business, Economics and Finance

Tue 29 Jul



In short:

The Commonwealth Bank of Australia has confirmed to the ABC that the organisation was cutting 45 jobs in its call centres after rolling out an artificial intelligence chatbot to deal with customer inquiries.

The move sparked outrage from the Finance Sector Union, which said impacted workers should be retrained and supported into new roles that leverage AI technology.

What's next?

A spokesperson for the bank said opportunities to redeploy or reskill staff would be prioritised.

abc.net.au/news/2025-07-29/commonwealth-bank-says-ai-behind-dozens-of-job-cuts/105586312

Commonwealth Bank backtracks on AI job cuts, apologises for 'error' as call volumes rise

By business reporter Stephanie Chalmers

AI

Thu 21 Aug



In short:

Commonwealth Bank announced 45 job cuts last month, as it introduced an AI "voice-bot", but has now reversed its decision.

The bank has apologised to affected employees for the "error", but the Finance Sector Union says the "damage is already done" to workers.

What's next?

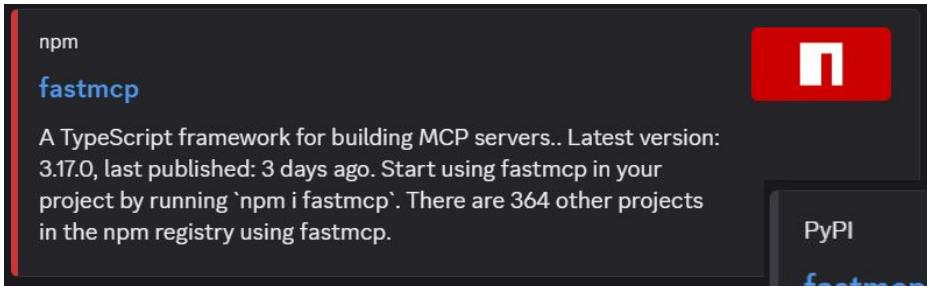
CBA says workers can choose to stay in their roles, be redeployed or leave the bank. Another Fair Work hearing is scheduled for next week.

abc.net.au/news/2025-08-21/cba-backtracks-on-ai-job-cuts-as-chatbot-lifts-call-volumes/105679492

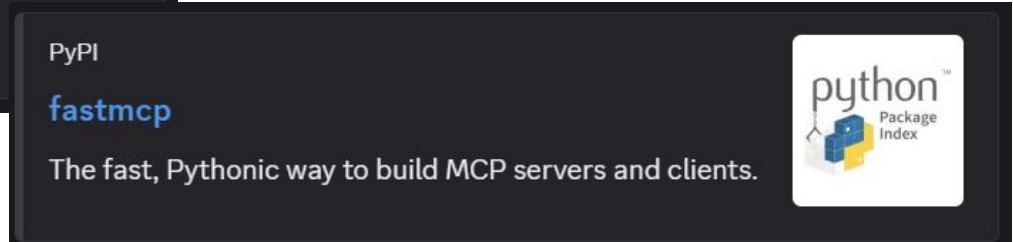
So, how do you make this stuff?

Creating an MCP Server

- Registering tools, resources, and prompts is just like handling events
- FastMCP is your friend - and available in Python & NodeJS



The screenshot shows the npm registry page for the `fastmcp` package. It features a red header bar with the npm logo. Below it, the package name `fastmcp` is displayed in blue. A brief description follows: "A TypeScript framework for building MCP servers.. Latest version: 3.17.0, last published: 3 days ago. Start using fastmcp in your project by running `'npm i fastmcp'`. There are 364 other projects in the npm registry using fastmcp." To the right of the npm section is a PyPI card.



The PyPI card for `fastmcp` includes the Python logo and the text "python Package Index". The package name is shown in blue, and the tagline "The fast, Pythonic way to build MCP servers and clients." is displayed below it.

Creating an MCP Server

```
const FastMCP = require("fastmcp").FastMCP;
const z = require("zod");  69k (gzipped: 15.1k)

const mcpServer = new FastMCP({
  name: "Super Cool MCP Server",
  version: "1.0.0"
});

mcpServer.addTool({
  name: "ping",
  description: "Basic sample tool that responds 'pong' to a given 'ping'.",
  execute: () => {
    return "pong"
  }
});

mcpServer.start({
  transportType: "stdio"
});
```

Adapting a REST API to Create an MCP server

- Organise your code to be D.R.Y, make functions where possible
- Remember to group functionality into as few tools as possible

```
router.get("/random", async (request, response) => {
  let result = await getRandomPokemon();
  response.json({
    name: result.name
  });
});

router.get("/id/:id", async (request, response) => {
  let result = await getPokemonByNumber(request.params.id);
  response.json({
    name: result.name
  });
});
```



```
mcpServer.addTool({
  name: "getRandomPokemon",
  description: "Selects a random Pokemon ID number and",
  execute: async () => {
    let result = await getRandomPokemon();
    return JSON.stringify(result);
  }
});

mcpServer.addTool({
  name: "getPokemonByID",
  description: "Fetches data about a specified Pokemon",
  parameters: z.object({
    id: z.number()
  }),
  execute: async (args) => {
    let result = await getPokemonByNumber(args.id);
    return JSON.stringify(result);
  }
});
```

Adapting a REST API to Create an MCP server

- Organise your code to be D.R.Y, make functions where possible
- Remember to group functionality into as few tools as possible

Avoid granular tools, but
make configurable tools!

```
    < mcpServer.addTool({
        name: "getAmountOfRandomPokemon",
        description: "Selects a specified quantity of random Pokem
        parameters: z.object({
            quantity: z.number()
        }),
        execute: async (args) => {
            let result = [];
            for (let index = 0; index < args.quantity; index++) {
                let retrievedData = await getRandomPokemon();
                result.push(retrievedData);
            }
            return JSON.stringify(result);
        }
    });
}
```

Get an MCP Client to Use an MCP server

1. Publish the MCP server to somewhere, such as:

- a. A package registry, such as NPM or PyPi
- b. A container registry, such as Docker Hub or GitHub Container Registry
- c. A private container registry that your private MCP client has access to
- d. One of the above, plus publish an entry in the MCP Registry

2. Configure the MCP client to be aware of the MCP server.

- a. Client settings file
- b. Client settings page
- c. VSCode workspace ".vscode/mcp.json" file

3. Use the MCP client and see what tools are detected.

Get an MCP Client to Use an MCP server

1. Publish the MCP server to somewhere, such as:
 - a. A package registry, such as NPM or PyPi
 - i. NPM packages need a “bin” command and “main” entry point configured
 - b. A container registry, such as Docker Hub or GitHub Container Registry
 - i. The container image should either:
 1. Expose a port for HTTP MCP communication
 2. Use logging appropriately for STDIO MCP communication
 - c. A private container registry that your private MCP client has access to
 - i. DevOps teams do Option B above with their wicked DevOps skills
 - d. One of the above, plus publish an entry in the MCP Registry

Get an MCP Client to Use an MCP server

1. Publish the MCP server to somewhere, such as:

- a. A package registry, such as NPM or PyPi
 - i. NPM packages need a “bin” command and “main” entry point configured

```
{  
  "name": "@alexstormwood/express-mcp-demo",  
  "version": "1.0.0",  
  "description": "Demo of how an ExpressJS server can be used as a LLM MCP server.",  
  "main": "src/mcp-server/index.js",  
  "bin": {  
    "express-mcp-demo": "./src/mcp-server/index.js"  
  },  
  ▷ Debug  
  "scripts": {  
    "start": "npm run start:api",  
    "start:api": "node src/web-api-server/index.js",  
    "start:mcp": "node src/mcp-server/index.js"  
  },  
}
```

Get an MCP Client to Use an MCP server

- b. A container registry, such as Docker Hub or GitHub Container Registry
 - i. The container image should either:
 1. Use logging appropriately for STDIO MCP communication
 2. Expose a port for HTTP MCP communication

```
mcpServer.start({  
  |   transportType: "stdio"  
});
```

```
ARG PORT=3456  
ENV PORT=${PORT}  
  
# Tell the image what to run when it starts.  
CMD ["npm", "run", "start:mcp"]  
  
# Expose the port that the server runs on.  
EXPOSE ${PORT}
```

```
mcpServer.start({  
  |   transportType: "httpStream",  
  |   httpStream: {  
  |     |   port: process.env.PORT || 3456  
  |   }  
});
```

Get an MCP Client to Use an MCP server

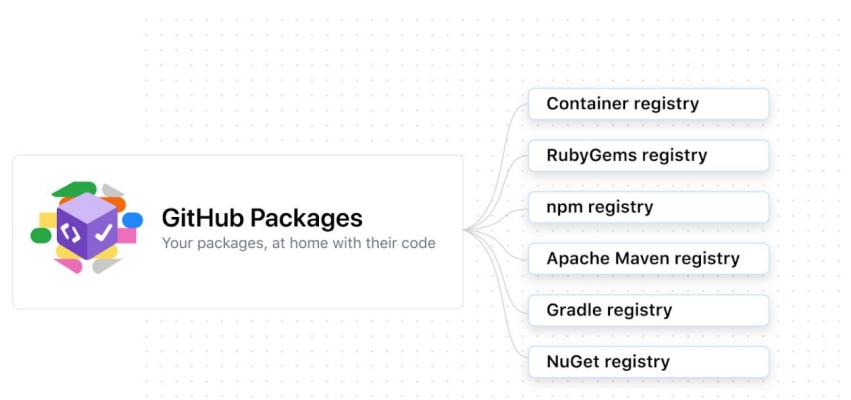
- c. A private container registry that your private MCP client has access to
 - i. DevOps teams do Option B above with their wicked DevOps skills

Get started with Amazon Elastic Container with AWS Free Tier →

Amazon Elastic Container Registry

Easily store, share, and deploy your container software anywhere

Get started with Amazon ECR



Get an MCP Client to Use an MCP server

- d. One of the above (previous steps), plus publish an entry in the MCP Registry

The image shows two screenshots of the MCP Registry GitHub repository. The top screenshot is a blog post titled "Introducing the MCP Registry" dated September 8, 2025. It discusses the launch of the MCP Registry, an open catalog and API for publicly available MCP servers. The bottom screenshot shows the repository's main page with two pull requests visible: one by rdimitrov to update documentation for remote MCP servers and another by domomegg to simplify GitHub Actions publishing. A section titled "Automate Publishing with GitHub Actions" provides instructions for setting up automated publishing using GitHub Actions.

mcp blog ☀️

September 8, 2025

Home » Posts

Introducing the MCP Registry

September 8, 2025 · 4 min · David Soria Parra (Lead Maintainer), Adam Jones (Registry Maintainer), Tadas Antanavicius (Registry Maintainer), Toby Padilla (Registry Maintainer), Theodora Chu (MCP PM at Anthropic)

Today, we're launching the Model Context Protocol (MCP) Registry—an open catalog and API for publicly available MCP servers to improve discoverability and implementation. By

modelcontextprotocol / registry

Code Issues 41 Pull requests 7 Discussions Actions

main registry / docs / guides / publishing / publish-server.md

rdimitrov Update the docs with info for remote MCP servers (#444) 🚧 ✓

Preview Code Blame

modelcontextprotocol / registry

Code Issues 41 Pull requests 7 Discussions Actions Security

main registry / docs / guides / publishing / github-actions.md

domomegg docs: simplify GitHub Actions publishing guide (#446) 🚧 ✓

Preview Code Blame

Publish Your MCP Server

Complete guide to publishing an MCP server to the registry.

Automate Publishing with GitHub Actions

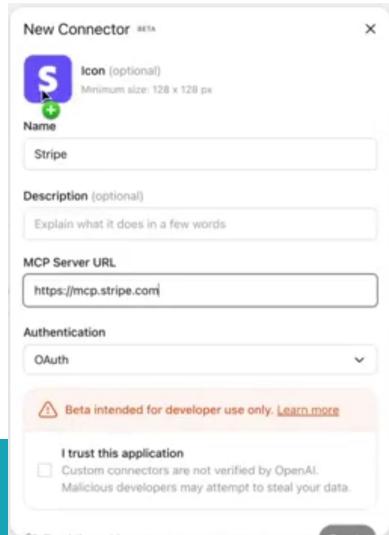
Set up automated MCP server publishing using GitHub Actions.

github.com/modelcontextprotocol/registry/tree/main/docs/guides/publishing

Get an MCP Client to Use an MCP server

2. Configure the MCP client to be aware of the MCP server.

- a. Client settings file
- b. Client settings page
- c. VSCode workspace ".vscode/mcp.json" file



```
.vscode > {} mcp.json > ...
1  {
2    "servers": {
3      "express-mcp-demo": {
4        "type": "stdio",
5        "command": "npm",
6        "args": [
7          "run",
8          "start:mcp"
9        ]
10       },
11     },
12     "inputs": []
13 }
```

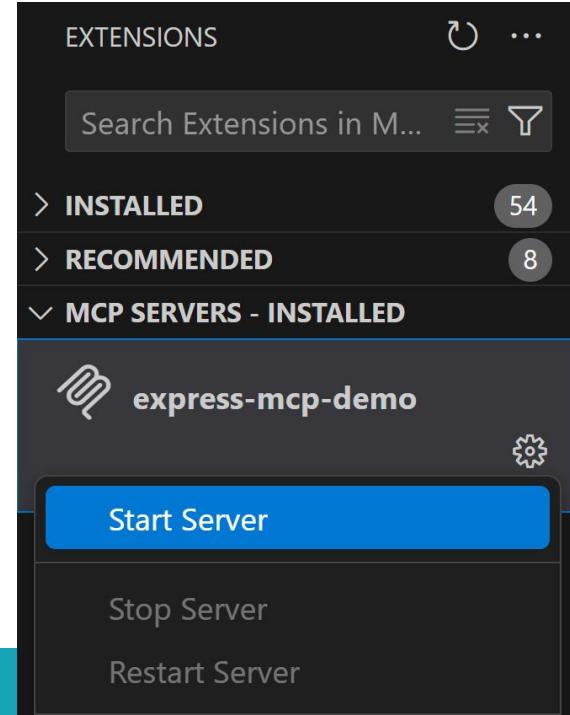
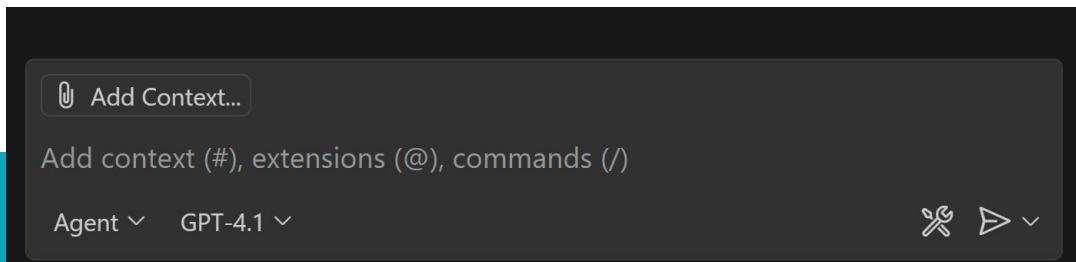
```
.vscode > {} mcp.json > ...
1  {
2    "servers": {
3      "express-mcp-demo": {
4        "command": "npx",
5        "args": [
6          "-y",
7          "published-npm-package-name"
8        ]
9      },
10     },
11     "inputs": []
12 }
```

Get an MCP Client to Use an MCP server

3. Use the MCP client and see what tools are detected.

For local MCP server development & testing:

- Set the VSCode Chat window to “Agent” mode
- Click on the tools icon near the bottom right to see all available tools
- Start your MCP server through the Extensions

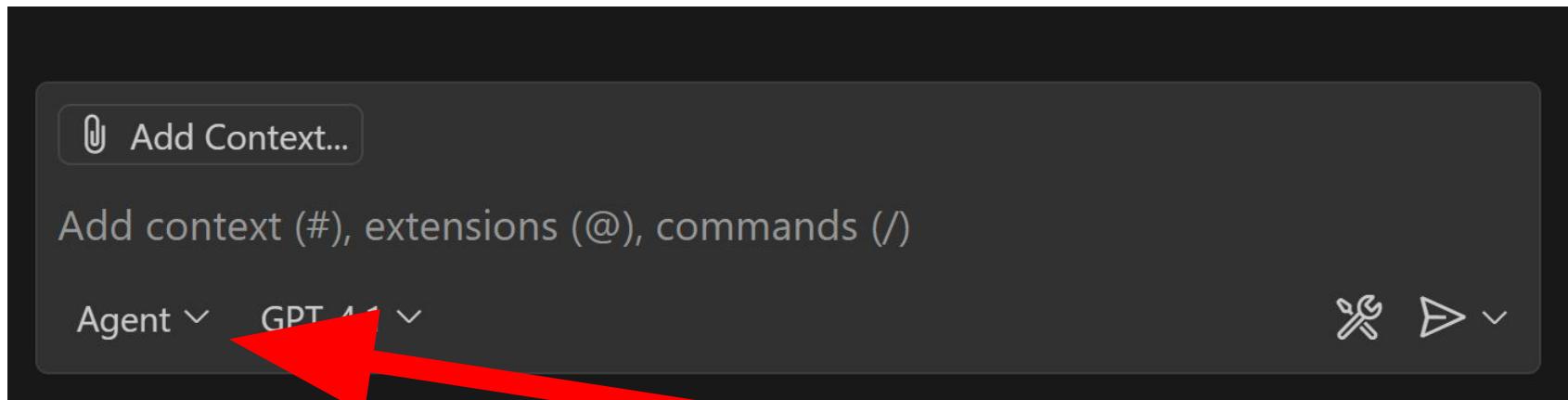


Get an MCP Client to Use an MCP server

3. Use the MCP client and see what tools are detected.

For local MCP server development & testing:

- Set the VSCode Chat window to “Agent” mode

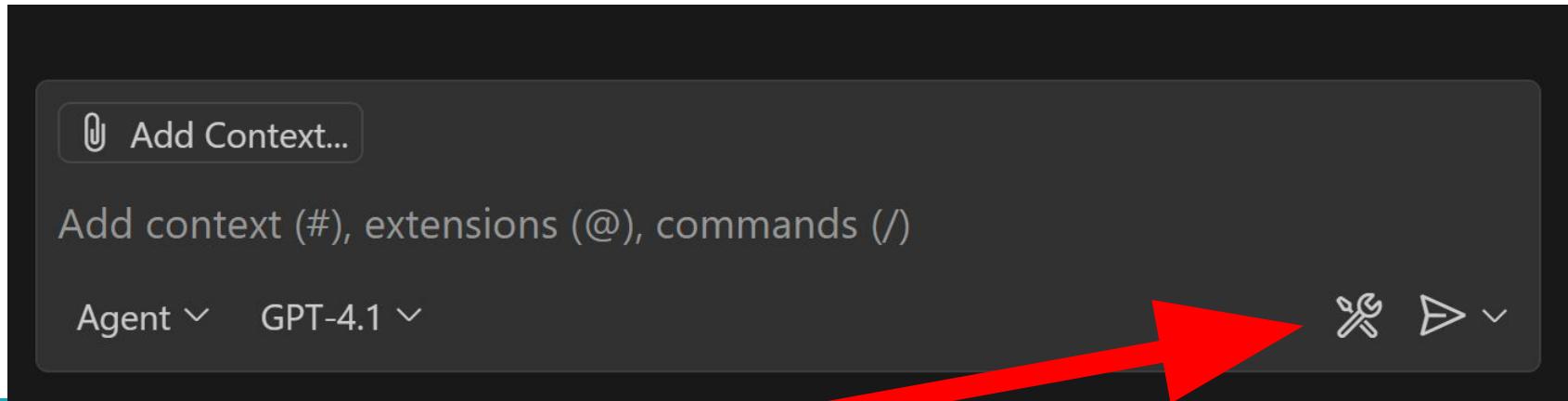


Get an MCP Client to Use an MCP server

3. Use the MCP client and see what tools are detected.

For local MCP server development & testing:

- Click on the tools icon near the bottom right to see all available tools



Get an MCP Client to Use an MCP server

3. Use the MCP client and see what tools are detected.

For local MCP server development & testing:

- Click on the tools icon near the bottom right to see all available tools

The screenshot shows a dark-themed MCP Client interface. At the top, there's a header bar with a dropdown arrow, a checked checkbox icon, a tools icon (represented by a wrench and screwdriver), the text "MCP Server: express-mcp-demo", and a gear icon for settings. Below the header is a list of detected tools, each with a checked checkbox, a tool icon (a stylized wrench), the tool name, and a brief description. The tools listed are: getAmountOfRandomPokemon, getPokemonByID, getRandomPokemon, and ping.

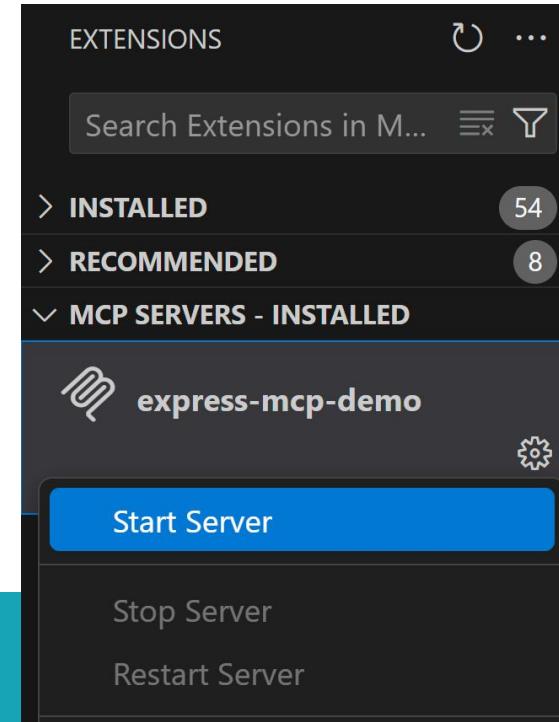
Tool	Description
getAmountOfRandomPokemon	Selects a specified quantity of random Pokemon ID number...
getPokemonByID	Fetches data about a specified Pokemon from the PokeAPI, where the user ...
getRandomPokemon	Selects a random Pokemon ID number and fetches data about that Pok...
ping	Basic sample tool that responds 'pong' to a given 'ping'.

Get an MCP Client to Use an MCP server

3. Use the MCP client and see what tools are detected.

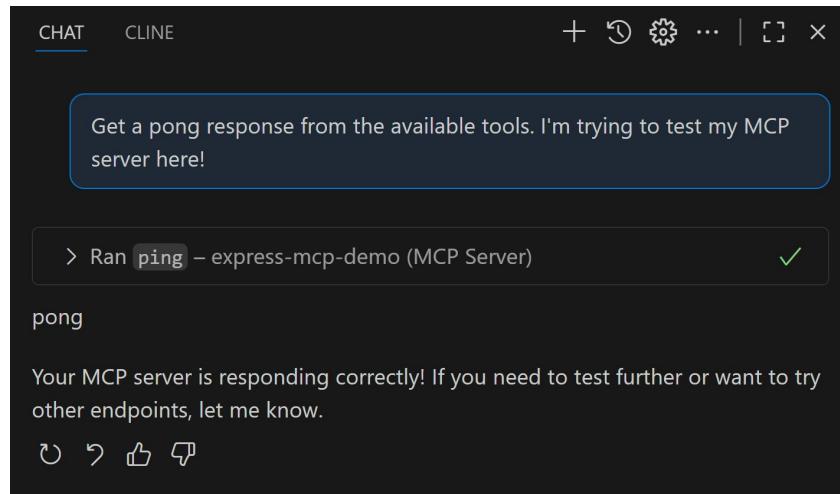
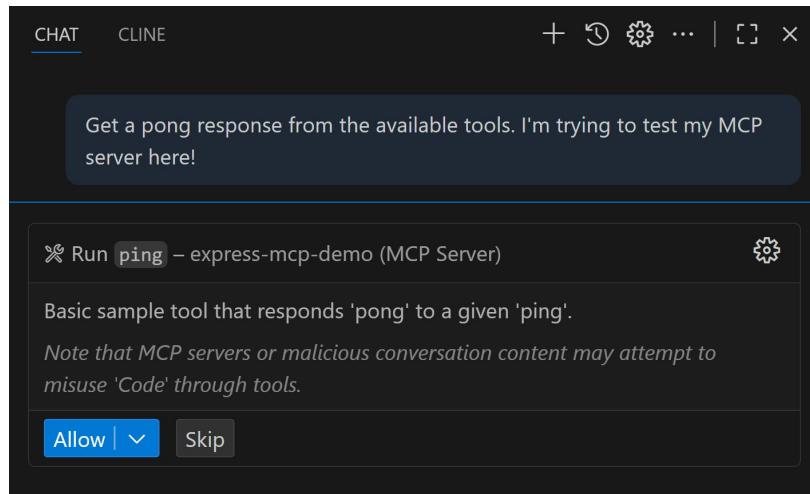
For local MCP server development & testing:

- Start your MCP server through the Extensions menu



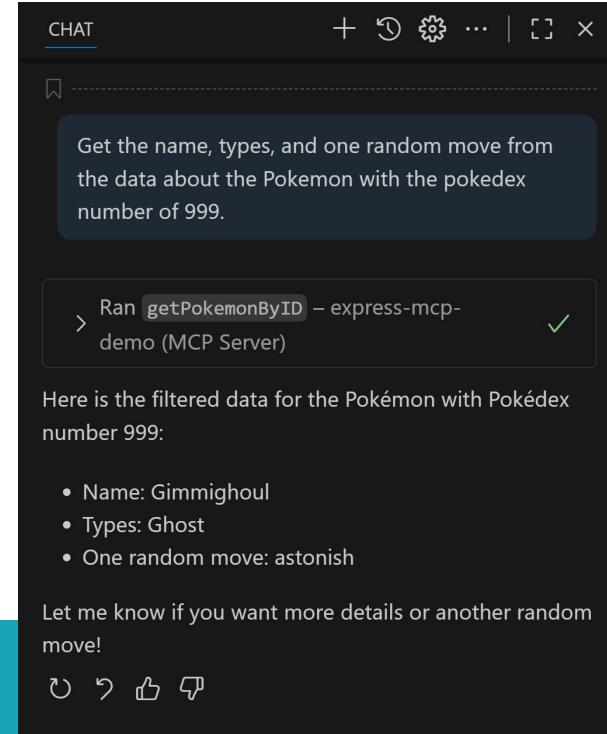
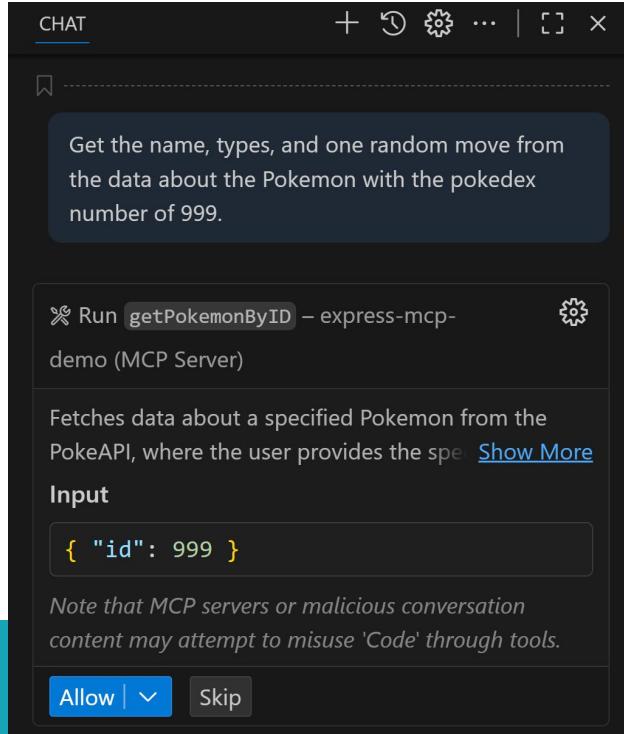
Get an MCP Client to Use an MCP server

3. Use the MCP client and see what tools are detected.



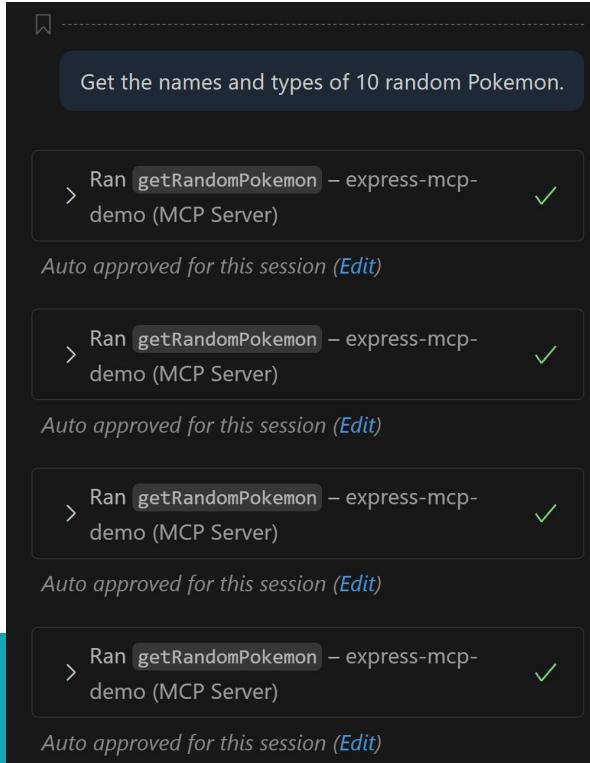
Get an MCP Client to Use an MCP server

3. Use the MCP client and see what tools are detected.



Get an MCP Client to Use an MCP server

3. Use the MCP client and see what tools are detected.



Bad!
Do not do this!

**Huge server responses
pile up, and
overwhelm the LLM
context window.**

Get an MCP Client to Use an MCP server

3. Use the MCP client and see what tools are detected.

Get the names and types of 10 random Pokémon.

Run `getAmountOfRandomPokemon` – express-mcp-demo (MCP Server)

Selects a specified quantity of random Pokémon ID numbers and fetches data about those P [Show More](#)

Input

```
{ "quantity": 10 }
```

Note that MCP servers or malicious conversation content may attempt to misuse 'Code' through tools.

Allow | Skip

Ran `getAmountOfRandomPokemon` – express-mcp-demo (MCP Server) ✓

✓ Summarized conversation history

The response for 10 random Pokémon is too large to display in a single message and is being truncated. To show you the full list of names and types for all 10 Pokémon, I recommend fetching and displaying them in smaller batches (e.g., 5 at a time) or by splitting the output into multiple messages.

Would you like to see the first 5 Pokémon's names and types, then the next 5? Or do you want the full raw JSON data in a downloadable file? Let me know your preference!

Show the first 5, and then the next 5.

Here are the names and types of the first 5 Pokémon from your batch request:

1. Inkay — Types: Dark, Psychic
2. Lairon — Types: Steel, Rock
3. Gothorita — Types: Psychic
4. Amaura — Types: Rock, Ice
5. Gurdurr — Types: Fighting

Would you like to see the next 5?

Yes please

The next 5 Pokémon and their types are:

6. Sableye — Types: Dark, Ghost
7. Dedenne — Types: Electric, Fairy
8. Toxicroak — Types: Poison, Fighting
9. Froslass — Types: Ice, Ghost
10. Pidgeotto — Types: Normal, Flying

**MCP usage in development is straightforward,
you run the server by following
your MCP client's instructions**

**MCP usage in production is
the wild west,
you only get complete control and
customisation by making
your own MCP clients**

Upcoming MCP Features & Developments

Highlights:

- Agent-to-tool authentication
- Protocol version compatibility negotiation
- MCP client security requirements

The screenshot shows a digital dashboard titled "SEP Review Pipeline". At the top, there's a header with a globe icon, the title "SEP Review Pipeline", a "View 1" dropdown, and a "Discard" button. Below the header is a search bar labeled "Filter by keyword or by field". The main area is divided into three vertical columns:

- Review Batch (4)**: Contains four items, each with a green circle icon and a user profile picture:
 - modelcontextprotocol #1288: WebSocket Transport (SEP-1288: Tool Filtering with Groups and Tags)
 - modelcontextprotocol #1300: Tool Filtering with Groups and Tags (SEP-1300: Specification Version Management)
 - modelcontextprotocol #1309: Specification Version Management (SEP-1330: Elicitation Enum Schema Improvements and Standards Compliance)
 - modelcontextprotocol #1330: Elicitation Enum Schema Improvements and Standards Compliance
- Accepted (6)**: Contains six items, each with a blue circle icon and a user profile picture:
 - modelcontextprotocol #1036: URL Mode Elicitation for secure out-of-band interactions (SEP-1036 : Support default values for all primitive types in elicitation schemas)
 - modelcontextprotocol #1034: Support default values for all primitive types in elicitation schemas (SEP-1046: Support OAuth client credentials flow in authorization)
 - modelcontextprotocol #973: Expose additional metadata for Implementations, Resources, Tools and Prompts (SEP-985: Align OAuth 2.0 Protected Resource Metadata with RFC 9728)
 - modelcontextprotocol #985: Align OAuth 2.0 Protected Resource Metadata with RFC 9728 (SEP-1024: MCP Client Security Requirements for Local Server Installation)
 - modelcontextprotocol #973: Expose additional metadata for Implementations, Resources, Tools and Prompts
 - modelcontextprotocol #985: Align OAuth 2.0 Protected Resource Metadata with RFC 9728
- Final (1)**: Contains one item, each with a green circle icon and a user profile picture:
 - modelcontextprotocol #994: Shared Communication Practices/Guidelines (SEP-994: Shared Communication Practices/Guidelines)

<https://github.com/modelcontextprotocol/modelcontextprotocol/projects?query=is%3Aopen>

Upcoming MCP Features & Developments

Highlights:

- Better community and official front-ends for discovering MCP servers on the “official” MCP server registry



Today, we’re launching the Model Context Protocol (MCP) Registry—an open catalog and API for publicly available MCP servers to improve discoverability and implementation. By

A screenshot of a browser window displaying the MCP Registry API endpoint at `registry.modelcontextprotocol.io/v0/servers`. The page shows a JSON response with one server entry. The JSON structure includes fields like '\$schema', 'name', 'description', 'status', 'repository', 'version', and 'remotes'.

```
{
  "servers": [
    {
      "$schema": "https://static.modelcontextprotocol.io/schemas/2025-07-09/server.schema.json",
      "name": "ai.waystation/gmail",
      "description": "Read emails, send messages, and manage labels in your Gmail account.",
      "status": "active",
      "repository": {
        "url": "https://github.com/waystation-ai/mcp",
        "source": "github"
      },
      "version": "0.3.1",
      "remotes": [
        {
          "type": "streamable-http",
          "url": "https://waystation.ai/gmail/mcp"
        }
      ]
    }
  ]
}
```

Top right: <https://blog.modelcontextprotocol.io/posts/2025-09-08-mcp-registry-preview/>

Bottom right: <https://registry.modelcontextprotocol.io/v0/servers>

Live Demo

github.com/AlexStormwood/express-mcp-demo

Thank you for attending!

Links and code in:

github.com/AlexStormwood/express-mcp-demo



alex@bigfootds.com

alex.holder@coderacademy.edu.au

alexstormwood.com

linkedin.com/in/alexstormwood