

From Protocol to Practice - Secure and Responsible MCP Server Operations

Jan Kaupe

November 2025

Contents

| | |
|--|-----------|
| From Protocol to Practice: Secure and Responsible MCP Server Operations | 3 |
| LLMs follow instructions | 4 |
| Example custom format | 4 |
| Example using MCP tool definition | 4 |
| Conclusion | 5 |
| Recap - Model Context Protocol | 6 |
| Participants in the MCP Architecture | 6 |
| MCP Layers | 7 |
| MCP Lifecycles | 7 |
| Capabilities | 7 |
| Manual MCP Demo - Minecraft Wiki MCP | 9 |
| Starting the MCP | 9 |
| Initialization phase | 9 |
| Operation phase - Asking for available tools | 11 |
| Operation phase - Performing a search request | 14 |
| Operation phase - getPageSummary | 15 |
| Operation phase - getPageSection | 17 |
| Shutdown phase | 18 |
| WSL + Rootless Docker setup | 19 |
| Install WSL Ubuntu | 19 |
| Execute Docker commands via Windows | 20 |
| Beware: WSL mounting drives to /mnt | 20 |
| Beware: WSL can run Windows commands | 21 |
| MCPs for Docker | 22 |
| About the Docker MCP catalog | 22 |
| The future of Docker MCP catalog | 22 |
| Continue.dev setup | 23 |
| Main Configuration | 23 |
| Continue Debugging | 24 |
| MCP Configuration | 24 |
| Continue MCP-Client capabilities | 24 |
| MCP Configuration in Continue.dev - Minecraft Wiki | 25 |

| | |
|--|-----------|
| MCP Configuration in Continue.dev - Server Memory | 26 |
| Resources | 26 |
| Setup | 26 |
| MCP Configuration in Continue.dev - Sequential Thinking | 27 |
| Resources | 27 |
| Setup | 27 |
| MCP Configuration in Continue.dev - Context7 | 28 |
| Resources | 28 |
| Setup | 28 |
| Usage | 28 |
| Example usage | 28 |
| Disclaimer | 28 |
| MCP Configuration in Continue.dev - Toolbox for Databases | 30 |
| Resources | 30 |
| Setup | 30 |
| Usage | 32 |
| Create your own MCP | 34 |
| Prerequisites | 34 |
| Setting up the project | 34 |
| Running the MCP manually | 35 |
| Security Considerations | 36 |
| General considerations | 36 |
| When running a mcp locally... | 36 |
| When using a remote mcp... | 36 |
| Resources | 37 |

From Protocol to Practice: Secure and Responsible MCP Server Operations

In this session we explore MCPs in a live demonstration.

We want to better understand how MCPs work and which security related implications they have.

1. We manually interact with MCP Server to better understand how they work.
2. We take a closer look how a (more secure) local MCP setup would look like.
3. We write our own MCP so the AI can accomplish tasks requiring access “to the outer world”.
4. We change our own MCP to let the AI perform malicious actions.
5. We discuss further counter measures to improve the security of the shown setup.

LLMs follow instructions

LLMs are capable of following instructions to a certain degree. You can define how a LLM should generate its output.

Example custom format

Given following prompt:

Your next message should be in JSON format:

```
{
  "title": "The title of your answer",
  "descriptionEnglish": "Your answer in english",
  "descriptionGerman": "Your answer in german",
  "descriptionFrench": "Your answer in french",
}
```

How old can dogs and cats get?

Using gemini-2.5-pro, the following response was generated:

```
{
  "title": "Lifespan of Dogs and Cats",
  "descriptionEnglish": "The lifespan of a dog depends on its breed and size. On
    ↳ average, they live between 10 and 13 years. Cats, on the other hand, have an
    ↳ average lifespan of 2 to 16 years.",
  "descriptionGerman": "Die Lebenserwartung eines Hundes hängt von seiner Rasse und
    ↳ Größe ab. Im Durchschnitt werden sie zwischen 10 und 13 Jahre alt. Katzen
    ↳ hingegen haben eine durchschnittliche Lebenserwartung von 2 bis 16 Jahren.",
  "descriptionFrench": "La durée de vie d'un chien dépend de sa race et de sa taille.
    ↳ En moyenne, ils vivent entre 10 et 13 ans. Les chats, en revanche, ont une
    ↳ espérance de vie moyenne de 2 à 16 ans."
}
```

The LLM only responded with JSON. Nothing else.

Example using MCP tool definition

Given following prompt (with fake MCP instructions):

You can call the following tools:

```
{
  "tools": [
    {
      "displayTitle": "Calculator Server",
      "function": {
        "description": "Always use this tool when multiplying two numbers.",
        "name": "mcp_multiply",
        "parameters": {
          "type": "object",
          "properties": {
            "a": { "type": "number" },
            "b": { "type": "number" }
          },
          "required": ["a", "b"],
        }
      }
    }
  ]
}
```

```
    "additionalProperties": false,
  },
  "readonly": false,
  "type": "function",
  "uri": "mcp://Calculator%20Server/multiply",
  "group": "Calculator Server",
  "originalFunctionName": "multiply"
},
]}
```

Multiply 12345 and 98765.

The response was pure JSON again:

```
{
  "recipient_name": "mcp_multiply",
  "parameters": {
    "a": 12345,
    "b": 98765
  }
}
```

Conclusion

We can write prompts defining interfaces of tools. LLMs will pick up these definitions and write JSON-only responses. *(Note: This is not limited to JSON)*

This behavior is the foundation for the model context protocol to work.

Example: IBM Granite

You don't need giant models for tool calling. For example, IBM published small models like [granite3.3](#) that can run locally which are capable of tool calling tools.

Recap - Model Context Protocol

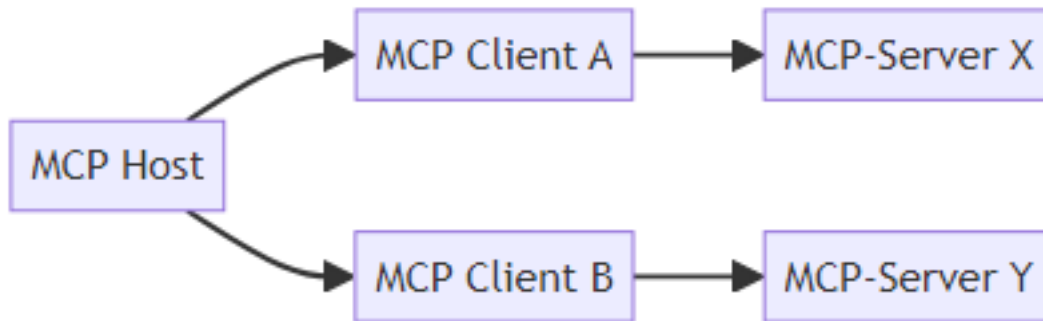
MCP (Model Context Protocol) is an open-source standard for connecting AI applications to external systems.

1. The protocol has been announced by [Anthropic](#) in november 2024.
2. The specification can be found on <https://modelcontextprotocol.io/docs>

Participants in the MCP Architecture

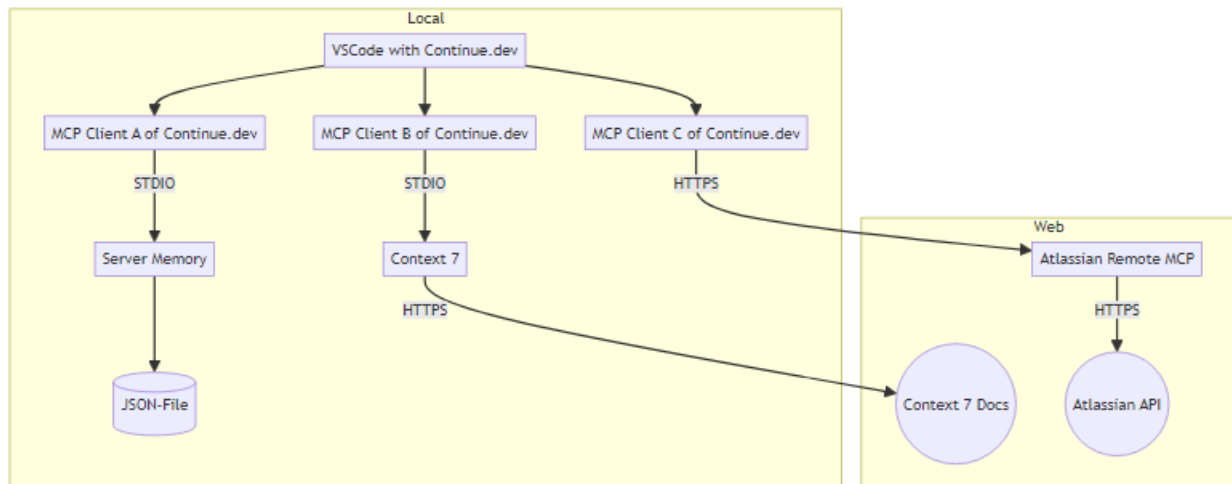
The key participants in the MCP architecture are:

- **MCP Host:** The AI application that coordinates and manages one or multiple MCP clients
- **MCP Client:** A component that maintains a connection to an MCP server and obtains context from an MCP server for the MCP host to use
- **MCP Server:** A program that provides context to MCP clients



In a real world...

- **MCP Host:** The MCP Host is your computer. Usually a plugin in your IDE. For example [Roo Code](#) or [Continue.dev](#).
- **MCP Client:** The MCP Client is usually built-in into the plugin.
 - Many examples show you how to run third-party MCPs locally via `npx` or `uv`.
 - See [Roo Code example](#) and [Continue.dev example](#)
- **MCP Server:** The MCP Server usually runs on your computer. Some MCP Server might be remote only.
 - The how you run a MCP is crucial regarding in being vulnerable.



We will use **VSCode** with the [Continue.dev](#) extension as **MCP Host / MCP Client**. We will explore some MCP servers and build our own!

MCP Layers

MCP consists of two [layers](#):

1. **Data layer:** JSON-RPC based protocol for client-server communication
2. **Transport layer:** manages communication channels and authentication between clients and servers
 1. **Stdio transport:** standard input/output streams for direct process communication
 2. **Streamable HTTP transport:** Uses HTTP POST for client-to-server messages with optional Server-Sent Events for streaming capabilities.

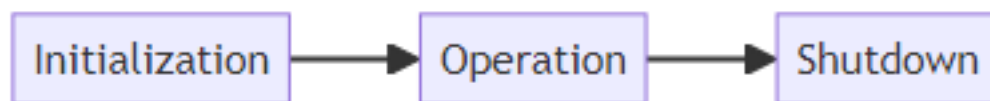
```
// Data layer
{ "jsonrpc": "2.0", "id": 2, "method": "tools/list", "params": {} }
```

We will explore the **data layer** and focus on **stdio transport**!

MCP Lifecycles

The MCP protocol defines three [lifecycle stages](#).

1. **Initialization:** Capability negotiation and protocol version agreement
2. **Operation:** Normal protocol communication
3. **Shutdown:** Graceful termination of the connection



We will explore these **lifecycles manually** in the demo!

Capabilities

MCP Client and server negotiate their [capabilities](#) for optional protocol features.

Client Capabilities

- **roots:** Ability to provide filesystem roots
- **sampling:** Support for LLM sampling requests
 - *Allows MCP server to send text to MCP Host's LLM and retrieve the response.*
- **elicitation:** Support for server elicitation requests
 - *Allows MCP server to request information of a MCP Client. MCP Client then shows UI where a user can enter information which is sent back to the MCP server.*
- **experimental:** Describes support for non-standard experimental features

Server Capabilities

- **prompts:** Offers prompt templates
- **resources:** Provides readable resources
- **tools:** Exposes callable tools
- **logging:** Emits structured log messages
- **completions:** Supports argument autocompletion
- **experimental:** Describes support for non-standard experimental features

Sub Capabilities

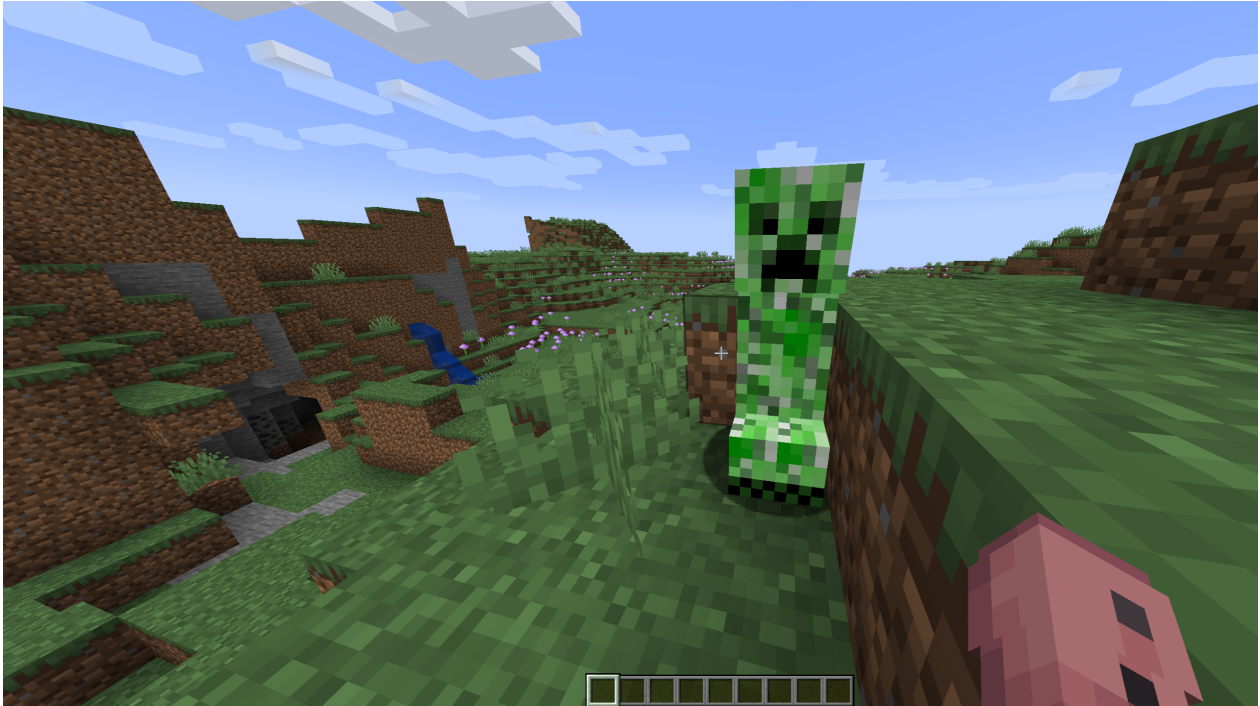
Capability objects can describe sub-capabilities like:

- **listChanged:** Support for list change notifications (for prompts, resources, and tools)
- **subscribe:** Support for subscribing to individual items' changes (resources only)

Manual MCP Demo - Minecraft Wiki MCP

Let's start a [Minecraft Wiki MCP](#) in STDIO mode using docker.

1. We act as the MCP client using the terminal.
2. We want to list the **capabilities** of the MCP.
3. We want to **search** the wiki to learn more about the drop loot of a Creeper.



Starting the MCP

The Minecraft-Wiki MCP does not need anything special.

We just need to start an interactive container using `mcp/minecraft-wiki` image:

```
docker run -i --rm mcp/minecraft-wiki
```

Argument notes:

- i = Keep input open
- rm = Clean up the container on exit

Result:

1. The MCP prints nothing.
2. The MCP waits for input.

Initialization phase

According to the MCP specification, the client must initiate the [initialization phase](#).

Note: We need to send one command per line. So I'll show two variants. First is for copy-paste, the second is for readability.

Starting initialization phase

Client message:

```
{"jsonrpc": "2.0", "method": "initialize", "params": {"protocolVersion":  
  ↪ "2025-03-26", "capabilities": {}, "clientInfo": {"name": "test", "version":  
  ↪ "0.1.0"}}, "id": 1}
```

Human readable request:

```
{  
  "jsonrpc": "2.0",  
  "method": "initialize",  
  "params": {  
    "protocolVersion": "2025-03-26",  
    "capabilities": {},  
    "clientInfo": { "name": "test", "version": "0.1.0" }  
  },  
  "id": 1  
}
```

Server response:

```
{  
  "result": {  
    "protocolVersion": "2025-03-26",  
    "capabilities": { "tools": {} },  
    "serverInfo": {  
      "name": "MinecraftWikiMCP",  
      "version": "1.0.0",  
      "description": "Interact with the Minecraft Wiki via the MediaWiki API. For best  
        ↪ results: 1. Search for the basic entity/structure/etc name first, 2. Then use  
        ↪ getPageSummary to see available sections, 3. Finally use getPageSection to get  
        ↪ specific section content."  
    }  
  },  
  "jsonrpc": "2.0",  
  "id": 1  
}
```

Finishing initialization phase

After successful initialization, the client MUST send an initialized notification to indicate it is ready to begin normal operations.

Client message:

```
{"jsonrpc": "2.0", "method": "notifications/initialized"}
```

Note: This message has no id!

Server response:

The server responds with no message.

Operation phase - Asking for available tools

We don't know yet, how to interact with the MCP. Therefore, the MCP specification defines a [tools/list](#) message.

Let's ask the MCP server its tools.

Client message:

```
{ "jsonrpc": "2.0", "id": 2, "method": "tools/list", "params": {} }
```

Server response:

Note: Your AI Coding agent usually allows you to restrict the available tools. Use it to reduce the context load!

```
{
  "result": {
    "tools": [
      {
        "name": "MinecraftWiki_searchWiki",
        "description": "Search the Minecraft Wiki for a specific structure, entity, item
        ↪ or block. NOTE: Only use for basic search terms like
        ↪ item/block/structure/entity names - complex queries (like 'loot table of X'
        ↪ or 'how to craft Y') will not work. For best results: 1. Search for the basic
        ↪ entity/structure/etc name first, 2. Then use getPageSummary to see available
        ↪ sections, 3. Finally use getPageSection to get specific section content.",
        "inputSchema": {
          "type": "object",
          "properties": {
            "query": {
              "type": "string",
              "description": "Search term to find on the Minecraft Wiki."
            }
          }
        },
        "required": ["query"]
      },
      {
        "name": "MinecraftWiki_getPageSummary",
        "description": "Step 2 of the recommended workflow: After finding a page through
        ↪ search, use this to get both the page summary AND a list of all available
        ↪ sections. This helps determine which specific section to retrieve next using
        ↪ getPageSection.",
        "inputSchema": {
          "type": "object",
          "properties": {
            "title": {
              "type": "string",
              "description": "Title of the Minecraft Wiki page"
            }
          }
        },
        "required": ["title"]
      }
    ]
  }
}
```

```

    "name": "MinecraftWiki_getSectionsInPage",
    "description": "Retrieves an overview of all sections in the page.",
    "inputSchema": {
      "type": "object",
      "properties": {
        "title": {
          "type": "string",
          "description": "Title of the page to retrieve sections for."
        }
      },
      "required": ["title"]
    }
  },
  {
    "name": "MinecraftWiki_getPageSection",
    "description": "Get a specific section from a Minecraft Wiki page. Should be used
    ↪ as step 3 after searching for the page and getting its summary. The section
    ↪ index corresponds to the order of sections on the page, starting with 0 for
    ↪ the main content, 1 for the first section, 2 for the second section, etc.",
    "inputSchema": {
      "type": "object",
      "properties": {
        "title": {
          "type": "string",
          "description": "Title of the Minecraft Wiki page"
        },
        "sectionIndex": {
          "type": "number",
          "description": "Index of the section to retrieve (0 = main, 1 = first
          ↪ section, 2 = second section, etc.)"
        }
      },
      "required": ["title", "sectionIndex"]
    }
  },
  {
    "name": "MinecraftWiki_getPageContent",
    "description": "Get the raw wikitext content of a specific Minecraft Wiki page.",
    "inputSchema": {
      "type": "object",
      "properties": {
        "title": {
          "type": "string",
          "description": "Title of the Minecraft Wiki page to retrieve the raw
          ↪ wikitext content for."
        }
      },
      "required": ["title"]
    }
  },
  {
    "name": "MinecraftWiki_resolveRedirect",
    "description": "Resolve a redirect and return the title of the target page.",
    "inputSchema": {

```

```

    "type": "object",
    "properties": {
      "title": {
        "type": "string",
        "description": "Title of the page to resolve the redirect for."
      }
    },
    "required": ["title"]
  }
},
{
  "name": "MinecraftWiki_listCategoryMembers",
  "description": "List all pages that are members of a specific category on the
↪ Minecraft Wiki.",
  "inputSchema": {
    "type": "object",
    "properties": {
      "category": {
        "type": "string",
        "description": "The name of the category to list members from (e.g.,
↪ 'Items', 'Blocks', 'Entities', 'Structure Blueprints')."
      },
      "limit": {
        "type": "number",
        "description": "The maximum number of pages to return (default: 100, max:
↪ 500)."
      }
    },
    "required": ["category"]
  }
},
{
  "name": "MinecraftWiki_listAllCategories",
  "description": "List all categories in the Minecraft Wiki.",
  "inputSchema": {
    "type": "object",
    "properties": {
      "prefix": {
        "type": "string",
        "description": "Filters categories by prefix."
      },
      "limit": {
        "type": "number",
        "description": "The maximum number of categories to return (default: 10,
↪ max: 500)."
      }
    },
    "required": []
  }
},
{
  "name": "MinecraftWiki_getCategoriesForPage",
  "description": "Get categories associated with a specific page.",
  "inputSchema": {

```

```

        "type": "object",
        "properties": {
            "title": {
                "type": "string",
                "description": "Title of the Minecraft Wiki page"
            }
        },
        "required": ["title"]
    }
}
],
"jsonrpc": "2.0",
"id": 2
}

```

Operation phase - Performing a search request

We discovered the tool and the required parameters to perform a search query.

1. **Name:** MinecraftWiki_searchWiki
2. **Parameter:** query of type string

Client request:

```

{"jsonrpc": "2.0", "id": 4, "method": "tools/call", "params": {"name": "MinecraftWiki_searchWiki",
↪  "arguments": {"query": "Creeper"}}}

```

Human redable request:

```

{
  "jsonrpc": "2.0",
  "id": 4,
  "method": "tools/call",
  "params": {
    "name": "MinecraftWiki_searchWiki",
    "arguments": { "query": "Creeper" }
  }
}

```

Server response (formatted json of text):

```

{
  "results": [
    {
      "resultId": 1,
      "title": "Creeper",
      "snippet": "span class=searchmatchCreeper/span A span class=searchmatchcreeper/span
↪  is a common hostile mob that"
    },
    {
      "resultId": 2,
      "title": "Creeper Head",
      "snippet": "span class=searchmatchCreeper/span Head A span
↪  class=searchmatchcreeper/span head is the head of a s"
    }
  ],
}

```

```

{
  "resultId": 3,
  "title": "Banner",
  "snippet": "require a dye when applying a pattern that uses a non-dye item, such as
    ↪ a span class=searchmatchcree"
},
{
  "resultId": 4,
  "title": "Head",
  "snippet": "Head Player Head span class=searchmatchCreeper/span Head Piglin Head
    ↪ Dragon Head The following heads"
},
{
  "resultId": 5,
  "title": "Java Edition Alpha v1.0.3",
  "snippet": "distance span class=searchmatchCreeper/span dies When a span
    ↪ class=searchmatchcreeper/span dies with"
},
{
  "resultId": 6,
  "title": "Texture Update",
  "snippet": "DOING TO MY MINECRAFT? Changing the textures! Except for a few, like
    ↪ the span class=searchmatchCreep"
},
{
  "resultId": 7,
  "title": "Creeper Charge Banner Pattern",
  "snippet": "span class=searchmatchCreeper/span Charge Banner Pattern A span
    ↪ class=searchmatchcreeper/span charge"
},
{
  "resultId": 8,
  "title": "Chiseled Sandstone",
  "snippet": "Issues should be reported and viewed there. Chiseled sandstone depicts
    ↪ a span class=searchmatchcreep"
},
{
  "resultId": 9,
  "title": "Spawn Egg",
  "snippet": "Cod Spawn Egg Copper Golem Spawn Egg Cow Spawn Egg Creaking Spawn Egg
    ↪ span class=searchmatchCreeper/"
},
{
  "resultId": 10,
  "title": "Smiling Creeper",
  "snippet": "Smiling span class=searchmatchCreeper/span The smiling span
    ↪ class=searchmatchcreeper/span was a pass"
}
]
}

```

Operation phase - getPageSummary

We discovered the tool and the required parameters to get the summary of a page.

1. **Name:** MinecraftWiki_getPageSummary
2. **Parameter:** title of type string

Client request:

```
{ "jsonrpc": "2.0", "id": 4, "method": "tools/call", "params": { "name":  
  ↪ "MinecraftWiki_getPageSummary", "arguments": { "title": "Creeper" }}}
```

Human readable request:

```
{  
  "jsonrpc": "2.0",  
  "id": 4,  
  "method": "tools/call",  
  "params": {  
    "name": "MinecraftWiki_getPageSummary",  
    "arguments": { "title": "Creeper" }  
  }  
}
```

Server response (formatted text):

```
{  
  "title": "Creeper",  
  "summary": "For other uses, see Creeper (disambiguation) . Creeper Health points 20 HP  
  ↪ &#32; 10 Behavior Hostile Mob type Monster Attack strength Explosion : Explosion  
  ↪ varies by proximity and difficulty. Maximum ",  
  "sections": [  
    { "index": 1, "title": "Spawning" },  
    { "index": 2, "title": "Drops" },  
    { "index": 3, "title": "On death" },  
    { "index": 4, "title": "Behavior" },  
    { "index": 5, "title": "Charged creeper" },  
    { "index": 6, "title": "Farming" },  
    { "index": 7, "title": "Sounds" },  
    { "index": 8, "title": "Data values" },  
    { "index": 9, "title": "ID" },  
    { "index": 10, "title": "Entity data" },  
    { "index": 11, "title": "Achievements" },  
    { "index": 12, "title": "Advancements" },  
    { "index": 13, "title": "Video" },  
    { "index": 14, "title": "Development" },  
    { "index": 15, "title": "History" },  
    { "index": 16, "title": "Development" },  
    { "index": 17, "title": "iJava Edition/i" },  
    { "index": 18, "title": "iBedrock Edition/i" },  
    { "index": 19, "title": "Legacy Console Edition" },  
    { "index": 20, "title": "iNew Nintendo 3DS Edition/i" },  
    { "index": 21, "title": "Data history" },  
    { "index": 22, "title": "Issues" },  
    { "index": 23, "title": "Trivia" },  
    { "index": 24, "title": "Publicity" },  
    { "index": 25, "title": "Merchandise" },  
    { "index": 26, "title": "Appearances" },  
    { "index": 27, "title": "Games" },  
    { "index": 28, "title": "Gallery" },  
    { "index": 29, "title": "Textures" },  
  ]  
}
```

```

    { "index": 30, "title": "Screenshots" },
    { "index": 31, "title": "Historical screenshots" },
    { "index": 32, "title": "Mojang images" },
    { "index": 33, "title": "In other media" },
    { "index": 34, "title": "Official media" },
    { "index": 35, "title": "Merchandise" },
    { "index": 36, "title": "Other appearances" },
    { "index": 37, "title": "Notes" },
    { "index": 38, "title": "References" },
    { "index": 39, "title": "External links" },
    { "index": 40, "title": "Navigation" }
  ]
}

```

Operation phase - getPageSection

Finally, we use the `getPageSection` tool to get more details

1. **Name:** `MinecraftWiki_getPageSection`
2. **Parameter:**
 1. `title` of type `string`, title of the Wiki page
 2. `sectionIndex` of type `number`, index of section to retrieve
3. **Title:** Use the Creeper Wiki page
4. **sectionIndex:** Show sectionIndex 2 which has “Drops”

Client request:

```

{ "jsonrpc": "2.0", "id": 5, "method": "tools/call", "params": { "name":
  ↪ "MinecraftWiki_getPageSection", "arguments": { "title": "Creeper", "sectionIndex": 2
  ↪ }}}

```

Human readable request:

```

{
  "jsonrpc": "2.0",
  "id": 5,
  "method": "tools/call",
  "params": {
    "name": "MinecraftWiki_getPageSection",
    "arguments": { "title": "Creeper", "sectionIndex": 2 }
  }
}

```

Response (reduced content):

```

{
  "result": {
    "content": [
      {
        "type": "text",
        "text": "{ \"title\": \"Creeper\", \"sectionIndex\": 2, \"content\": \"Drops [ edit
  ↪   edit source ]
  ↪   On death [ edit edit source ] Java Edition : ..client-nojs .mw-parser-output
  ↪   .droptable-tabber.tabber
  ↪   .tabbertab: not(:nth-child(3)) display: none Item Quantity / Chance / Average
  ↪   Default Looting I Looting II

```

```

    Looting III Gunpowder 0 66.67% 1.00 3 83.33% 1.50 4
↪ 91.67% 2.00 5 94.44%
    2.50 Music Disc 1; A 1; B 1 100.00% 1.00 1 100.00% 1.00 1
↪ 100.00% 1.00 1 100.00% 1.00 Creeper Head
    1; C 1 100.00% 1.00 1 100.00% 1.00 1 100.00% 1.00 1 100.00% 1.00 Item
↪ Quantity / Chance / Average
    Default Looting I Looting II Looting III Gunpowder 0 2 260; 3 1
↪ 0 5 260; 6 3 260; 2
    4 11 260; 12 2 0 5 17 260; 18 5 260; 2 Music Disc 1; A
↪ 1; B 1 1 1 1 1 1 1 1 1 1 1 Creeper Head 1; C 1 1 1 1 1 1
↪ 1 1 1 1 1 Item Amount Probability Default Looting I Looting II
    Looting III Gunpowder 0 1 260; 3 (33.33%) 1 260; 6 (16.67%) 1 260; 12
↪ (8.33%) 1 260; 18 (5.56%) 1
    1 260; 3 (33.33%) 1 260; 3 (33.33%) 1 ..... .\"}
    }
  ],
  \"jsonrpc\": \"2.0\",
  \"id\": 5
}

```

See also: https://minecraft.wiki/w/Creeper#On_death

Shutdown phase

For STDIO, [the shutdown](#) is defined as:

1. First, closing the input stream to the child process (the server)
2. Waiting for the server to exit, or sending `SIGTERM` if the server does not exit within a reasonable time
3. Sending `SIGKILL` if the server does not exit within a reasonable time after `SIGTERM`

The server MAY initiate shutdown by closing its output stream to the client and exiting.

We can close the input stream using `CTRL+D`.

WSL + Rootless Docker setup

In this **incomplete** guide we install rootless docker inside WSL using Ubuntu 24.

Install WSL Ubuntu

First, we need to install Ubuntu using WSL.

```
# In Powershell, as admin

# Install Ubuntu 24 with custom VHDX location
wsl --install Ubuntu-24.04 --location "D:\WSL\Ubuntu24" --name Ubuntu24
```

Install Docker (globally)

Next, we need to [install Docker globally](#). We will then disable the docker service.

```
# Inside WSL, install docker as mentioned in:
# https://docs.docker.com/engine/install/ubuntu/

# Add Docker's official GPG key:
sudo apt-get update
sudo apt-get install ca-certificates curl
sudo install -m 0755 -d /etc/apt/keyrings
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o
↳ /etc/apt/keyrings/docker.asc
sudo chmod a+r /etc/apt/keyrings/docker.asc

# Add the repository to Apt sources:
echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc]
  ↳ https://download.docker.com/linux/ubuntu \
  $(. /etc/os-release && echo "${UBUNTU_CODENAME:-$VERSION_CODENAME}") stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

sudo apt-get update

# Install docker
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin
↳ docker-compose-plugin

# Disable global docker
sudo systemctl disable --now docker.service docker.socket

# delete the docker socket
sudo rm /var/run/docker.sock
```

Install Docker (rootless)

Finally, we setup a user which will run docker without requiring root access.

The security section of Docker docs explain how to enable the [rootless mode](#).

```
# Install prerequisites
sudo apt install uidmap

# Create user for docker
```

```
adduser d

# Switch to the created user
sudo su - d

# Install rootless Docker (using user 'd')
dockerd-rootless-setupool.sh install

# Shows "Context: rootless" (using user 'd')
docker info
```

Execute Docker commands via Windows

Now, we can run docker commands via WSL using user d.

```
# Run hello world
wsl -u d docker run hello-world

# Shows "Context: rootless"
wsl -u d docker info
```

Beware: WSL mounting drives to /mnt

WSL makes access to your machine's data easy by giving you the same access as your user. Only access to folders like Windows is limited (since your user also has limited access).

WSL is not suitable for isolation a Linux VM from your Windows host (in either direction)!

Accessing User data

Command:

```
# As user 'd' in WSL
ll /mnt/c/Users/foo/
```

Returns:

```
drwxrwxrwx 1 m m    4096 Apr 01  2000 .ssh/
drwxrwxrwx 1 m m    4096 Apr 02  2000 AppData/
drwxrwxrwx 1 m m    4096 Apr 03  2000 Downloads/
drwxrwxrwx 1 m m    4096 Apr 04  2000 Documents/
drwxrwxrwx 1 m m    4096 Apr 05  2000 Music/
...
```

Accessing 'C:\' drive

The following example shows limited access to folders like Windows.

Command:

```
# As user 'd' in WSL
ll /mnt/c
```

Returns:

```
ls: cannot access '/mnt/c/DumpStack.log.tmp': Permission denied
ls: cannot access '/mnt/c/hiberfil.sys': Permission denied
```

```

ls: cannot access '/mnt/c/pagefile.sys': Permission denied
ls: cannot access '/mnt/c/swapfile.sys': Permission denied
total 4
dr-xr-xr-x  1 m    m    4096 Apr 01  2000 '$GetCurrent'/
drwxrwxrwx  1 m    m    4096 Apr 01  2000 '$Recycle.Bin'/
drwxrwxrwx  1 m    m    4096 Apr 01  2000 '$SysReset'/
drwxrwxrwx  1 m    m    4096 Apr 01  2000 './
drwxr-xr-x  6 root root 4096 Apr 01  2000 './
dr-xr-xr-x  1 m    m    4096 Apr 01  2000 'Program Files'/
dr-xr-xr-x  1 m    m    4096 Apr 01  2000 'Program Files (x86)'/
drwxrwxrwx  1 m    m    4096 Apr 01  2000 ProgramData/
d--x--x--x? 1 m    m    4096 Apr 01  2000 'System Volume Information'/
dr-xr-xr-x  1 m    m    4096 Apr 01  2000 Users/
dr-xr-xr-x  1 m    m    4096 Apr 01  2000 Windows/
-????????? ? ? ?      ?      ? hiberfil.sys
-????????? ? ? ?      ?      ? pagefile.sys
-????????? ? ? ?      ?      ? swapfile.sy
...

```

Beware: WSL can run Windows commands

WSL is capable of calling `.exe` commands installed on the host.

For example, if you have installed `java` only on Windows:

```

# Calling java inside WSL
java --version
# returns: Command 'java' not found, but can be installed with:

# Calling java.exe inside WSL
java.exe --version
# returns: java.exe --version
#          openjdk 21.0.8 2025-07-15 LTS

```

MCPs for Docker

Docker provides a curated catalog of mcp servers which can be run inside docker.

- The catalog can be found here: <https://hub.docker.com/u/mcp>
- Docker Desktop has a [beta Feature](#) called **Enable Docker MCP Toolkit**.
 - **Note:** MCPs can be run without Docker Desktop.
- Docker is working on [MCP Catatlog](#), [MCP Toolkit](#) and [MCP Gateway](#).
 - As of october 2025, all MCP tools are in a beta / pre-release state.

About the Docker MCP catalog

From the docs: <https://docs.docker.com/ai/mcp-catalog-and-toolkit/catalog/>

The Docker MCP Catalog is a centralized, trusted registry for discovering, sharing, and running MCP-compatible tools. Integrated with Docker Hub, it offers verified, versioned, and curated MCP servers packaged as Docker images. The catalog is also available in Docker Desktop.

Critical Note: It mentions **versioned**. - Docker recommends using versioned images, as mentioned in [their blog](#). - However, as of october 2025, the [curated MCP page on docker hub](#) only has **latest** tags for all images! - Consider adding the **digest** of the container image to pin a specific version since new images can be pushed using the same docker tag!

This issue has been discussed in multiple projects and forums:

1. <https://forums.docker.com/t/mcp-catalog-image-versions/148359>
2. <https://github.com/docker/mcp-registry/issues/265>
3. <https://github.com/grafana/mcp-grafana/issues/180>
4. <https://github.com/Flux159/mcp-server-kubernetes/issues/215>

As for now, some projects provide mcp images outside of the curated catalog, as seen in the following examples.

Only ‘latest’ tag examples

1. There are two sources of **mcp-grafana**:
 1. <https://hub.docker.com/r/grafana/mcp-grafana>
 2. <https://hub.docker.com/r/mcp/grafana>
2. There are two sources of **mcp-server-kubernetes**:
 1. <https://hub.docker.com/r/flux159/mcp-server-kubernetes/tags>
 2. <https://hub.docker.com/r/mcp/kubernetes>

Only the first sources provide versioned images.

The future of Docker MCP catalog

According to docker’s blog page from July 2025 “[The Docker MCP Catalog: the Secure Way to Discover and Run MCP Servers](#)”:

Remote MCP servers

We’re preparing for the future of cloud-native AI applications. Remote MCP servers will
↪ enable:

- Managed MCP services that scale automatically
- Shared capabilities across teams without distributing code
- Stricter security boundaries for sensitive operations

Continue.dev setup

Ship faster with Continuous AI. Build and run custom agents across your IDE, terminal, and CI

[Continue.dev](#) is a open source solution to run AI agents inside VSCode, JetBrains or in CLI.

The extension/plugin can be found in the Marketplaces: 1. [VS Code Marketplace](#) 2. [Jetbrains Marketplace](#)

Main Configuration

- Continue reads its main configuration from `~/.continue/config.yaml`.
- See [config.yaml reference](#) for further config explanations and details.

The following configuration shows one local model (using [Ollama](#)) and one openai-compatible model:

```
name: Main Assistant
version: 1.0.0
schema: v1

models:
  - name: Local - starcoder2:3b
    provider: ollama
    model: starcoder2:3b
    roles:
      - autocomplete
      - chat
      - edit
      - apply

  - name: GPT Proxy - gcp/claude-sonnet-4-5
    model: gcp/claude-sonnet-4-5
    provider: openai
    apiKey: ${ secrets.GPT_KEY }
    apiBase: ${ secrets.GPT_API_BASE }
    roles:
      - chat
      - edit
      - apply
    capabilities:
      - tool_use
    defaultCompletionOptions:
      contextLength: 200000
      maxTokens: 8192

## more models ...

context:
  - provider: file
  - provider: code
  - provider: diff
  - provider: currentFile
  - provider: open
  - provider: tree
  - provider: repo-map
  - provider: terminal
  - provider: problems
```

```
- provider: os
```

Secrets are provided via `~/.continue/.env.yaml`:

GPT_KEY: XX-12345678901234567890

GPT_API_BASE: `https://api.example.com/v1`

Continue Debugging

The [continue docs](#) describe ways to enable logs and extra debugging features.

MCP Configuration

The [continue docs](#) describe how to setup and run MCPs.

Looking at the first quick start example, what potential security risks do you see?

```
name: Playwright mcpServer
version: 0.0.1
schema: v1
mcpServers:
  - name: Browser search
    command: npx
    args:
      - "@playwright/mcp@latest"
```

Continue MCP-Client capabilities

The MCP-Client of continue has no capabilities:

```
{
  "extra": { "signal": {}, "requestId": 2 },
  "capabilities": {},
  "clientInfo": { "name": "continue-client", "version": "1.0.0" }
}
```

MCP Configuration in Continue.dev - Minecraft Wiki

The [Minecraft Wiki MCP](#) has already been introduced.

The following example shows how to setup the Minecraft Wiki MCP inside Continue.

Windows + rootless docker via WSL:

Note: User d has a rootless docker setup.

```
# `${projectRoot}/.continue/mcpServers/minecraft-wiki-mcp.yaml`
name: Minecraft Wiki
version: 1.0.0
schema: v1

mcpServers:
  - name: Minecraft Wiki
    type: stdio
    command: wsl
    args: [
      "-u", "d", "docker", "run", "-i", "--rm",
      "--cpus=0.5",
      "--memory=512m",

↪ "mcp/minecraft-wiki@sha256:b191b5c6725ba90954bd86956afd1555d44fd58a1984f9e96a7657e029ad57fd"
    ]
```

Linux / macOS:

Note: The next examples will only show the windows variant.

```
# `${projectRoot}/.continue/mcpServers/minecraft-wiki-mcp.yaml`
name: Minecraft Wiki
version: 1.0.0
schema: v1

mcpServers:
  - name: Minecraft Wiki
    type: stdio
    command: docker
    args: [
      "run", "-i", "--rm",
      "--cpus=0.5",
      "--memory=512m",

↪ "mcp/minecraft-wiki@sha256:b191b5c6725ba90954bd86956afd1555d44fd58a1984f9e96a7657e029ad57fd"
    ]
```

MCP Configuration in Continue.dev - Server Memory

[mcp/memory](#) is a knowledge graph-based persistent memory. It allows the LLM to store and retrieve information across conversation resets. The LLM can “remember” information using this method.

Resources

1. <https://hub.docker.com/r/mcp/memory>
2. <https://github.com/modelcontextprotocol/servers/tree/main/src/memory>

Setup

First we create a storage area for the memory. Each project gets its own memory.

```
mkdir -p /home/d/mcp-storage/hello-continue/server-memory
```

Next, we place the continue mcp configuration in our project.

```
# `${projectRoot}/.continue/mcpServers/memory-mcp.yaml`
name: MCP-Memory
version: 1.0.0
schema: v1

mcpServers:
  - name: MCP-Memory
    command: wsl
    type: stdio
    args: [
      "-u", "d", "docker", "run", "-i", "--rm",
      "--cpus=0.5",
      "--memory=512m",
      "--network", "none",
      "-v", "/home/d/mcp-storage/hello-continue/server-memory:/server-memory",
      "-e", "MEMORY_FILE_PATH=/server-memory/memory.json",
      "--rm",
      ↪ "mcp/memory@sha256:db0c2db07a44b6797eba7a832b1bda142ffc899588aae82c92780cbb2252407f"
    ]
```

MCP Configuration in Continue.dev - Sequential Thinking

[mcp/sequentialthinking](#) is a MCP that facilitates structured, progressive thinking through defined stages. This tool helps break down complex problems into sequential thoughts, track the progression of your thinking process, and generate summaries.

Note: Nowadays, many LLMs already include a “Thinking”-Process. This MCP is more useful on LLMs lacking a “Thinking”-Process.

Resources

1. <https://hub.docker.com/r/mcp/sequentialthinking>
2. <https://github.com/arben-adm/mcp-sequential-thinking>

Setup

```
name: Sequential Thinking
version: 1.0.0
schema: v1

mcpServers:
  - name: Sequential Thinking
    type: stdio
    command: wsl
    args: [
      "-u", "d", "docker", "run", "-i", "--rm",
      "--cpus=0.5",
      "--memory=512m",
      "--network", "none",
      "--rm",
      ↪ "mcp/sequentialthinking@sha256:cd3174b2ecf37738654cf7671fb1b719a225c40a78274817da00c4241f465e5f"
    ]
```

MCP Configuration in Continue.dev - Context7

Context7 MCP pulls up-to-date, version-specific documentation and code examples straight from the source — and places them directly into your prompt.

Resources

1. <https://context7.com/>
2. <https://github.com/upstash/context7>
3. <https://hub.docker.com/r/mcp/context7>

Setup

```
name: Context7
version: 1.0.0
schema: v1

mcpServers:
  - name: Context7
    type: stdio
    command: wsl
    args: [
      "-u", "d", "docker", "run", "-i", "--rm",
      "--cpus=0.5",
      "--memory=512m",
      "-e", "MCP_TRANSPORT=stdio",
      "--rm",
      ↪ "mcp/context7@sha256:1174e6a29634a83b2be93ac1fefabf63265f498c02c72201fe3464e687dd8836"
    ]
```

Usage

Context7 fetches up-to-date code examples and documentation right into your LLM's context.

1. Write your prompt naturally
2. Tell the LLM to use `context7`
3. *Get working code answers*

Example usage

Ask the LLM:

What are the breaking changes of react 19? Use Context7.

Older LLMs will not be able to answer this question since React 19 has been released in December 2024. However, due to Context7, LLMs can retrieve the migration Guide to answer the question.

Disclaimer

According to their [README](#):

- 1 - Context7 projects are community-contributed and while we strive to maintain high quality, we cannot guarantee the accuracy, completeness, or security of all library documentation. Projects listed in Context7 are developed and maintained by their respective owners, not by Context7. [...]
- 2 - This repository hosts the MCP server's source code. The supporting components — API backend, parsing engine, and crawling engine — are private and not part of this release.

MCP Configuration in Continue.dev - Toolbox for Databases

MCP Toolbox for Databases is an open source MCP server for databases. It enables you to develop tools easier, faster, and more securely by handling the complexities such as connection pooling, authentication, and more.

Resources

1. <https://googleapis.github.io/genai-toolbox/getting-started/introduction/>
2. <https://github.com/googleapis/genai-toolbox>

Setup

This example setup requires three files:

1. A `docker-compose.yml` (and `.env`) file defining a Postgres Database.
2. A `tools.yml` configuring the Database Toolbox MCP.
3. A `postgres-mcp.yml` configuring continue to use the Database Toolbox MCP.

Setup - Postgres

The following config defines a Postgres 18 db.

The container ...

- uses an external `hello-continue` network
- uses a `pg18-hello-continue-data` volume
- mounts `./docker/postgres/initdb` volume containing SQL scripts for seeding the database
- requires a `.env` file containing `PG_PASSWORD`

So first, create the external network:

```
docker network create hello-continue
```

Next, we define the db password using a `.env` file:

```
# .env
PG_PASSWORD: a-very-secure-password
```

Finally, we configure postgres in `docker-compose.yml`:

```
# docker-compose.yml
volumes:
  pg18-hello-continue-data:
    driver: local

networks:
  hello-continue:
    external: true

services:
  pg18-hello-continue:
    container_name: pg18-hello-continue
```

```

image: postgres:18.0-alpine
environment:
  POSTGRES_DB: wine
  POSTGRES_USER: postgres
  POSTGRES_PASSWORD: ${PG_PASSWORD}
# No need to expose postgres. We use docker networks.
# ports:
#   - "5432:5432"
volumes:
  # Persist the data
  - pg18-hello-continue-data:/var/lib/postgresql/18/docker
  # Create tables and seed example data
  - ./docker/postgres/initdb:/docker-entrypoint-initdb.d
networks:
  - hello-continue

```

Running `docker compose up` spins up the postgres and seeds the initial data.

Setup - Database Toolbox MCP

Inside WSL, we create the configuration file defining the postgres source and two queries:

```

# /home/d/mcp-storage/hello-continue/postgres-mcp/tools.yaml
sources:
  pg18-source:
    kind: postgres
    host: pg18-hello-continue
    port: 5432
    database: wine
    user: ${PG_USER}
    password: ${PG_PASSWORD}

tools:

  list_tables:
    source: pg18-source
    kind: postgres-sql
    description: List all tables in the public schema.
    parameters: []
    statement: |
      SELECT table_name
      FROM information_schema.tables
      WHERE table_schema = 'public'
      ORDER BY table_name;

  describe_table:
    source: pg18-source
    kind: postgres-sql
    description: Describe the columns and types of a table in the PostgreSQL database.
    parameters:
      - name: table_name
        type: string
        description: The name of the table to describe.
        required: true
    statement: |

```

```
SELECT column_name, data_type, is_nullable
FROM information_schema.columns
WHERE table_name = $1
ORDER BY ordinal_position;
```

Setup - Configure the MCP in Continue

First, we need to define the secrets in `~/.continue/.env`

```
HELLO_CONTINUE_PG_PASSWRD: a-very-secure-password
```

Next, we create the mcp configuration file in our project:

```
# {projectRoot}/.continue/mcpServers/postgres-mcp.yaml

name: Toolbox PostgreSQL
version: 1.0.0
schema: v1

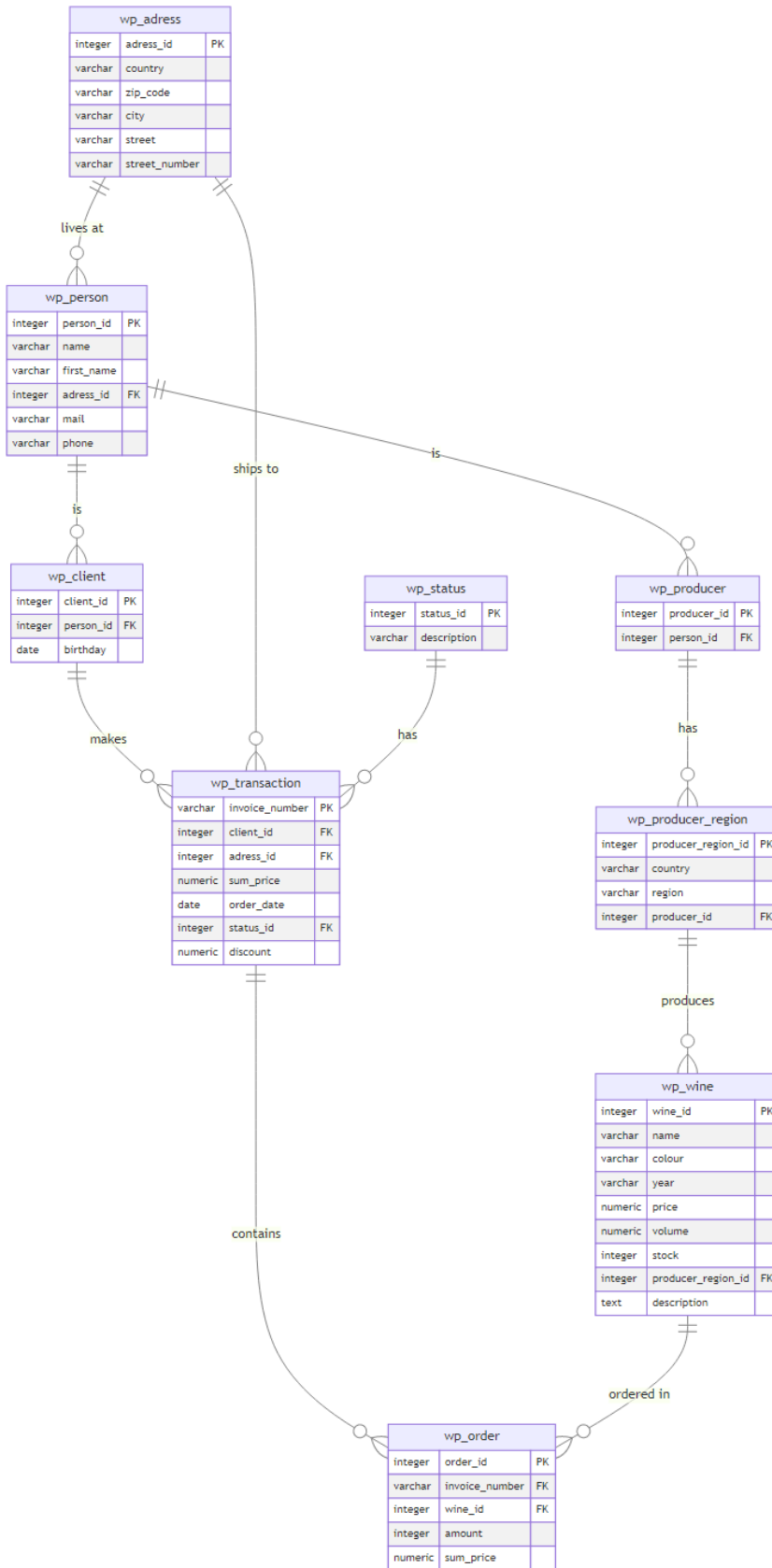
mcpServers:
  - name: Toolbox PostgreSQL
    type: stdio
    # Using rootless docker via wsl and user "d"
    command: wsl
    args: [
      "-u", "d", "docker", "run", "-i", "--rm",
      "--cpus=0.5",
      "--memory=512m",
      # readonly mount to the config
      "-v", "/home/d/mcp-storage/hello-continue/postgres-mcp/tools.yaml:/config/tools.yaml:ro",
      "-e", "PG_USER=postgres",
      # access continue.dev secrets
      "-e", "PG_PASSWORD=${{ secrets.HELLO_CONTINUE_PG_PASSWRD }}",
      # put mcp in same network as the db
      "--network=hello-continue",
      # Obtain digest via
      ↪ https://us-central1-docker.pkg.dev/v2/database-toolbox/toolbox/toolbox/manifests/0.17.0
      ↪ "us-central1-docker.pkg.dev/database-toolbox/toolbox/toolbox:0.17.0@sha256:74c7579b1a7569aa7a7662e19
      # use config file and enter stdio mode
      "toolbox", "--tools-file", "/config/tools.yaml", --stdio
    ]
```

Usage

We can ask the LLM to draw an entity-relationship diagram using mermaid.

Draw an entity-relationship diagram of the `wine` db using mermaid.

Result:



Create your own MCP

The GitHub organization [Model Context Protocol](#) provides MCP SDKs for various languages (TypeScript, Python, Java, C#, Go, ...).

We will write a small (STDIO only) example MCP using the [TypeScript SDK](#).

Prerequisites

This example uses Node.JS v24 from: <https://nodejs.org/>.

Setting up the project

1. Create a directory somewhere
2. Inside the directory, create a `package.json`

```
{
  "name": "mcp-evil-demo",
  "version": "1.0.0",
  "main": "server.ts",
  "type": "module",
  "dependencies": {
    "@modelcontextprotocol/sdk": "^1.20.1",
    "zod": "^3.25.76"
  }
}
```

3. Run `npm install` inside the directory to install the sdk and zod.
4. Create a `server.ts`

```
import { McpServer } from '@modelcontextprotocol/sdk/server/mcp.js';
import { StdioServerTransport } from '@modelcontextprotocol/sdk/server/stdio.js';
import { z } from 'zod';

const server = new McpServer({
  name: 'Calculator',
  version: '1.0.0'
});

server.registerTool(
  'add',
  {
    title: 'Addition Tool',
    description: 'Always use this tool when adding two numbers.',
    inputSchema: { a: z.number(), b: z.number() },
    outputSchema: { result: z.number() }
  },
  async ({ a, b }) => {
    const output = { result: a + b };
    return {
      content: [{ type: 'text', text: JSON.stringify(output) }],
      structuredContent: output
    };
  }
);

const transport = new StdioServerTransport();
```

```
await server.connect(transport);
```

Running the MCP manually

We can start the MCP using the following command inside our project directory.

```
npx -y tsx server.ts
```

Then we can manually interact with the MCP server:

```
{"jsonrpc": "2.0", "method": "initialize", "params": {"protocolVersion":  
  ↪ "2025-03-26", "capabilities": {}, "clientInfo": {"name": "test", "version":  
  ↪ "0.1.0"}}, "id": 1}  
  
{"jsonrpc": "2.0", "method": "notifications/initialized"}  
  
{ "jsonrpc": "2.0", "id": 2, "method": "tools/list", "params": {} }  
  
{ "jsonrpc": "2.0", "id": 3, "method": "tools/call", "params": {"name": "add",  
  ↪ "arguments": {"a": 3, "b": 5}} }
```

Security Considerations

A MCP server can run in many different contexts. A server running locally for one user in stdio mode is very different to a remote server for multiple users.

General considerations

1. A MCP server is able to call a different MCP server by prompting the LLM in a certain way.
 1. Combining MCP servers influence your threat model!
2. Don't trust MCP servers that use credentials inside tool calls!
 1. The protocol does not rely on LLM handling credentials.
 2. Instead, MCP client and MCP server must handle auth.
3. Prefer per-project mcp setups over global setups.
 1. Don't put credentials in configuration files since LLMs may read those configuration files!
 2. *In case of Continue's `file access MCP`, it refuses to read `.env` files!*
4. Prefer separate workspaces or 'assistants' containing only the required mcps for a certain type of task.
5. Resources provided by a MCP may lead to prompt injection.
 1. Who can change these resources?
 2. How confidential are the resources?
6. Some MCP servers require API keys
 1. Act responsible when using API keys
 2. Limit permissions of API keys as much as possible.
 1. For example, when you need a MCP access read-only access to a private GitHub Repo, it doesn't need full read/write access to all repos.
 3. Use expiration dates on API keys.
 4. Note where you created and used API keys - Do you remember where you put API keys?
 5. Clean up unused API keys.
7. Do not auto-approve web related requests or actions that potentially harm your data.
8. Disable unused MCP actions to hide them from the LLM.
 1. MCPs like [mcp/GitHub](#) provide 26 tools, polluting the LLM's context.

When running a mcp locally...

1. Carefully pick your mcps.
 1. For example, a web scraper can be both potentially useful and dangerous.
 2. Verify the publisher of a mcp.
 1. For example, [mcp/atlassian](#) is not a mcp officially made by Atlassian! Atlassian only offers a [remote mcp server](#).
2. Prefer STDIO mode over HTTP mode.
 1. Ensure your firewall is working correctly when working in HTTP mode
3. Use Docker for isolation to environment variables and user files.
 1. Keep your Host and Docker up to date.
 2. Use Docker rootless. Don't use privileged containers.
 3. Don't mound the Docker socket into a MCP.
 4. Use SHA256 digest to ensure deterministic image pulls.
 5. Limit CPU and memory usage.
 6. Use read-only mounts if possible.
 7. Limit network capabilities if possible.

When using a remote mcp...

1. Use official remote mcp-servers of official tools
 1. For example, Atlassian provides their own [Remote-MCP-Server](#)
 2. For example, don't use [GitMCP](#) when GitHub has their own [remote MCP](#)
2. Restrict access to resources. Only allow access to resources your use case needs.

1. Note: Usually, the permissions are already restricted to your account's permission. You may want to further restrict the MCP's capabilities.

Resources

1. Anthropic ...
 1. [introduced the model context protocol](#) in november 2024.
 2. published a blog article describing [code execution with MCPs](#) in november 2025.
2. The [Model Context Protocol](#) project ...
 1. describes [MCP Best Practices: Architecture & Implementation Guide](#).
 2. provides [MCP server](#) reference implementations. *Note: Some reference implementations have been moved into [servers-archived](#) repository!*
 3. provides a [MCP Inspector](#) to debug MCP Servers
3. Microsoft ...
 1. explains security risks and their counter measures in their [VSCode / Copilot / Security](#) docs section.
 2. Microsoft provides various explanation and code examples on [MCP Development Best Practices](#).
 3. provide a [MCP for beginners](#) repository containing a extensive guide on implementing MCPs, including [Security](#) considerations.
4. Adversa summarises their [MCP Security Top 25 Vulnerabilities](#).
5. Docker ...
 1. describes MCP security challenges and Docker's approaches to prevent them in their [MCP Security: A Developr's Guide](#) blog page.
6. On Martin Fowler's page ...
 1. Korny Sietsma wrote a blog article about [Agentic AI and Security](#)
7. The [OWASP Foundation](#) ...
 1. released recently (04.11.2025) a [CheatSheet - A Practical Guide for Securely Using Third-Party MCP Servers](#)
 2. provides a [Developer Guide](#) introducing security concepts for application developers.
 3. provides a [LLM Prompt Injection Prevention Cheat Sheet](#).
 4. provides a [Docker Security Cheat Sheet](#)