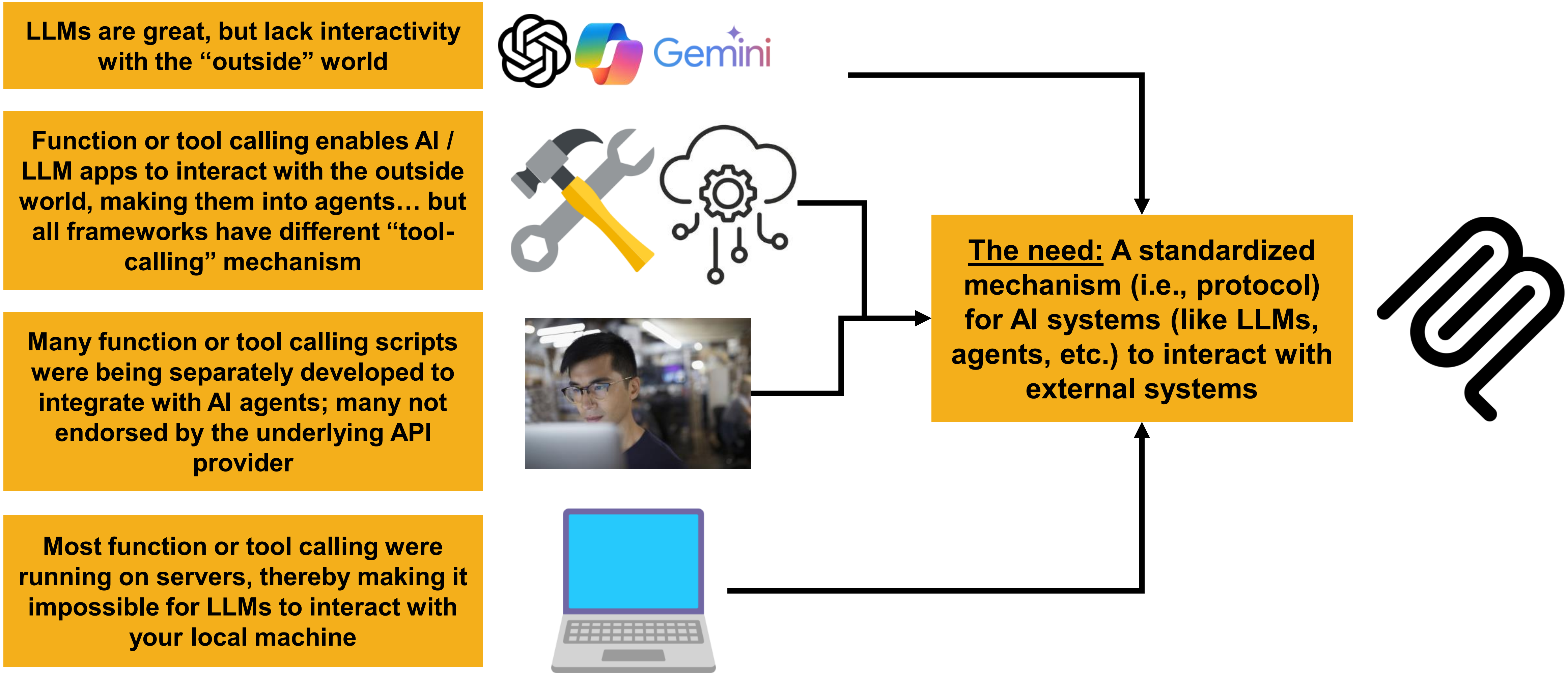
https://linktr.ee/henrylearning

The idea behind MCP



LLMs are great, but lack interactivity with the "outside" world

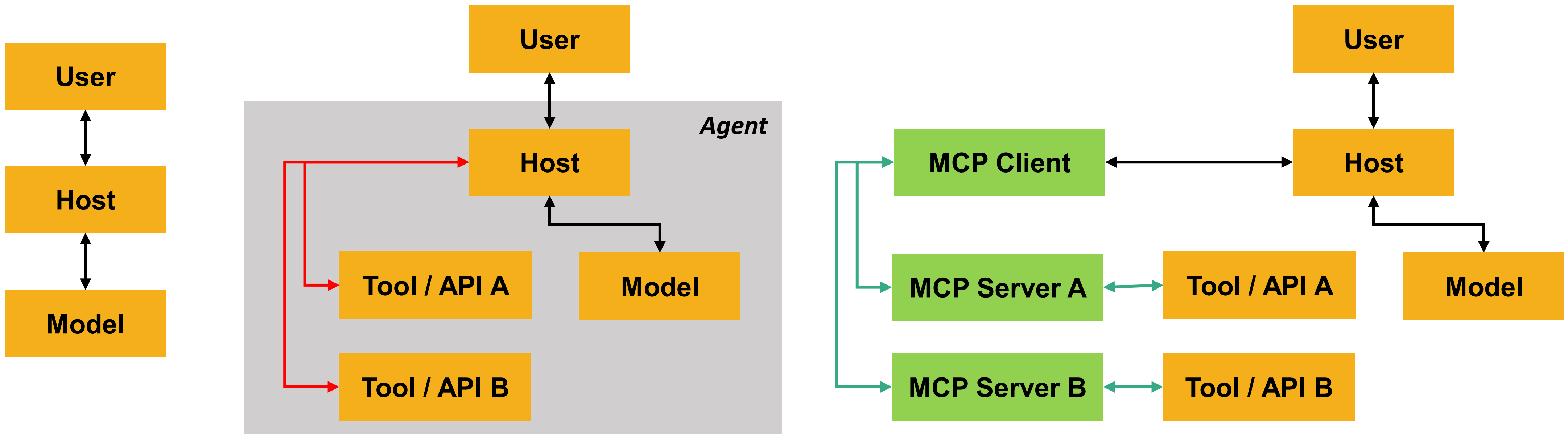
Function or tool calling enables AI / LLM apps to interact with the outside world, making them into agents… but all frameworks have different "tool-calling" mechanism

Many function or tool calling scripts were being separately developed to integrate with AI agents; many not endorsed by the underlying API provider

Most function or tool calling were running on servers, thereby making it impossible for LLMs to interact with your local machine

The need: A standardized mechanism (i.e., protocol) for AI systems (like LLMs, agents, etc.) to interact with external systems

History of Agentic AI Development / MCP



Why this is inefficient:

- Items in the red arrow had to be manually programmed every single time. Each time the "host" would change, or the tool / API would change, the connection needed to be updated and reprogrammed.
- Also, two separate projects that connected to "Tool A" would do it differently and it would be double the work. Why?

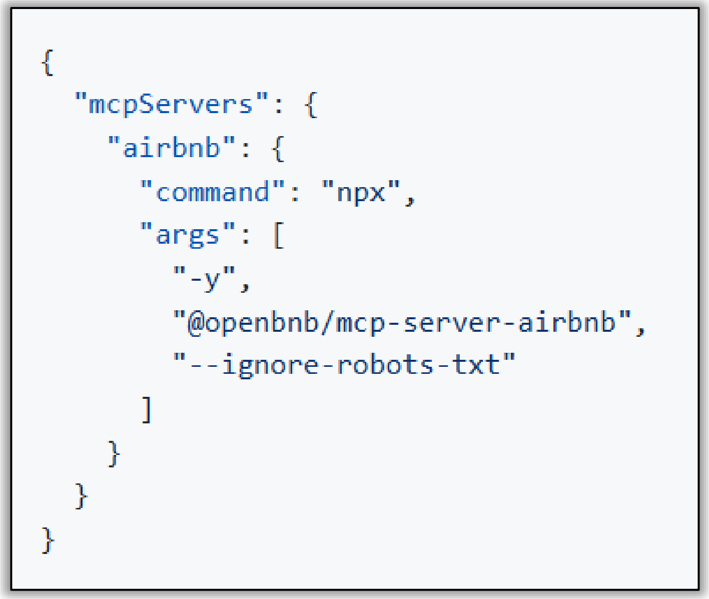
Why this is better:

- Items in the **green arrow** represents the standardized MCP protocol.
- Now, developers that create agentic AI apps only need to support to an "MCP client" and by doing so, automatically can connect to thousands of MCP servers with a few lines of JSON
- The green arrows are programmed once, updatable, standardized, and even supported by the underlying API providers (build once, run everywhere). Developers don't need to worry about the **green arrow**.

User and developer benefits

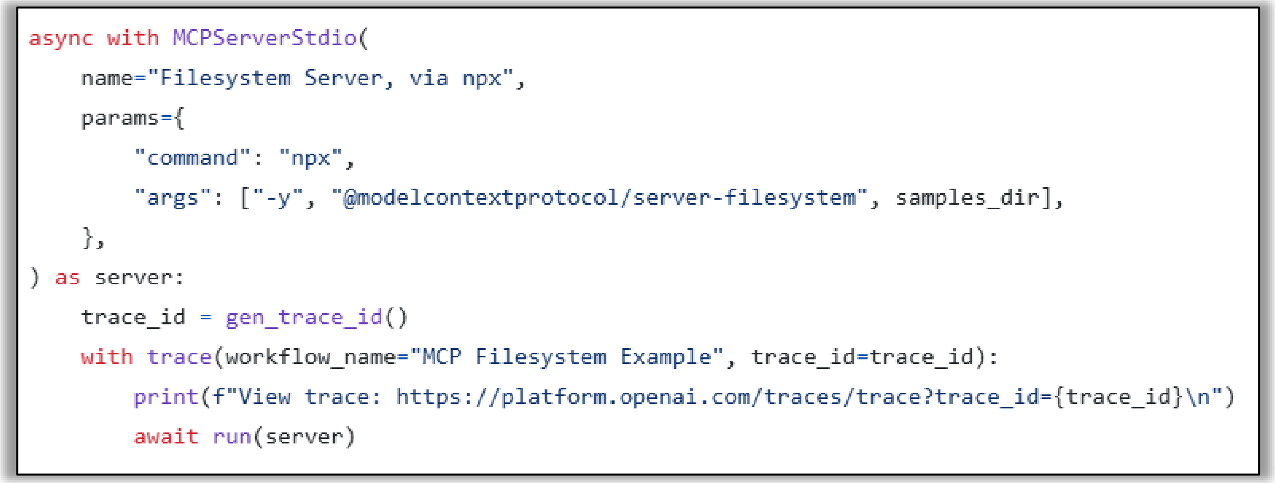
This makes it <u>extremely</u> simple for LLM host users and developers to connect to thousands of different external systems and tools, provided that… (1) host supports the “MCP client” and (2) the external system has an “MCP server”

User



```
{
  "mcpServers": {
    "airbnb": {
      "command": "npx",
      "args": [
        "-y",
        "@openbnb/mcp-server-airbnb",
        "--ignore-robots-txt"
      ]
    }
  }
}
```

Agent Developer

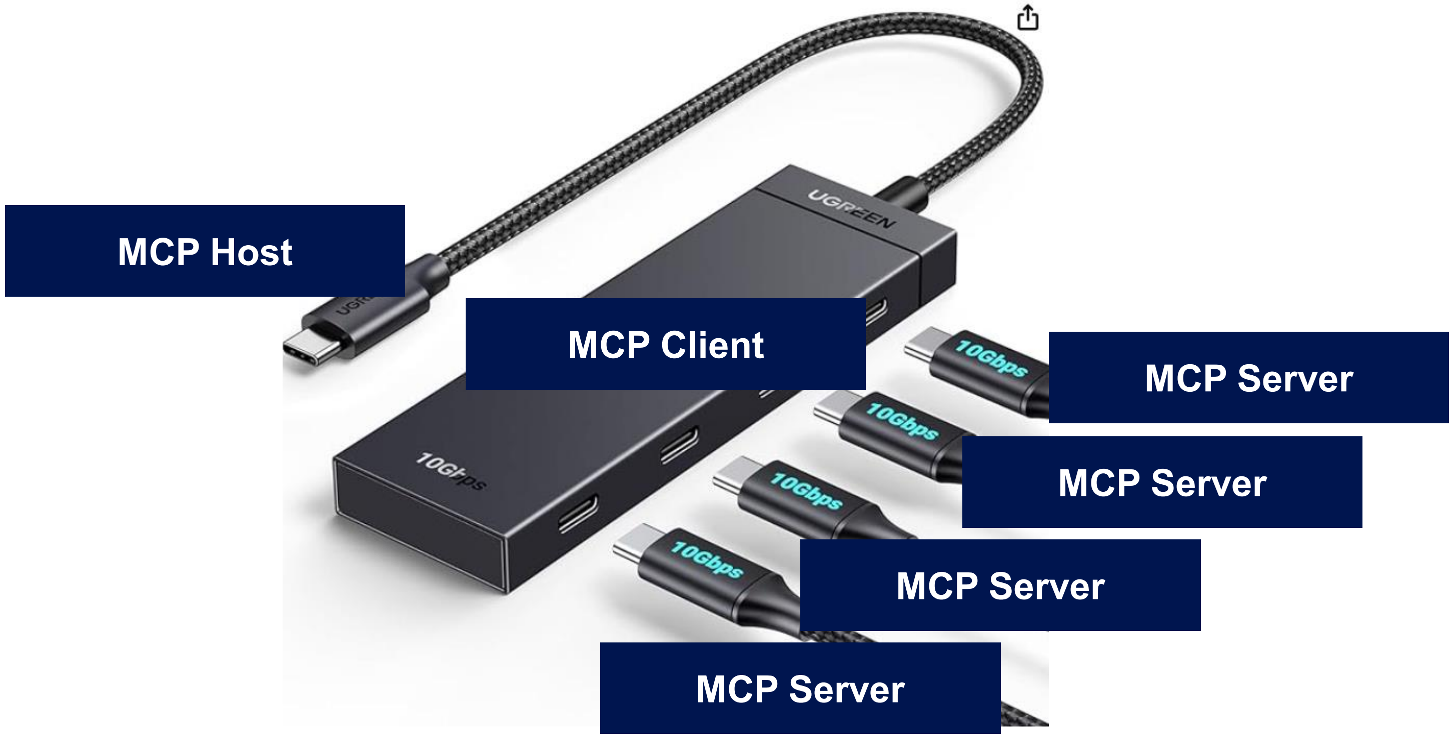


```
async with MCPServerStdio(
    name="Filesystem Server, via npx",
    params={
        "command": "npx",
        "args": ["-y", "@modelcontextprotocol/server-filesystem", samples_dir],
    },
) as server:
    trace_id = gen_trace_id()
    with trace(workflow_name="MCP Filesystem Example", trace_id=trace_id):
        print(f"View trace: https://platform.openai.com/traces/trace?trace_id={trace_id}\n")
        await run(server)
```

https://linktr.ee/henrylearning

What is MCP?

A standardized mechanism (i.e., protocol) for AI systems (like LLMs, agents, etc.) to interact with external systems (like APIs, tool logic, local processes, etc.)



MCP Host

MCP Client

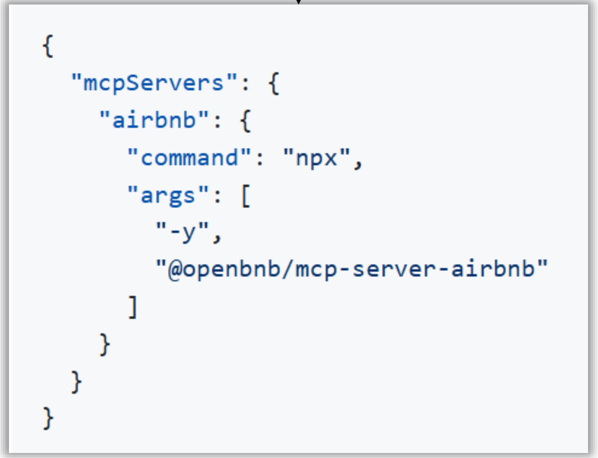
MCP Server

MCP Server

MCP Server

MCP Server

Hundreds of lines of code every single time you want a new tool

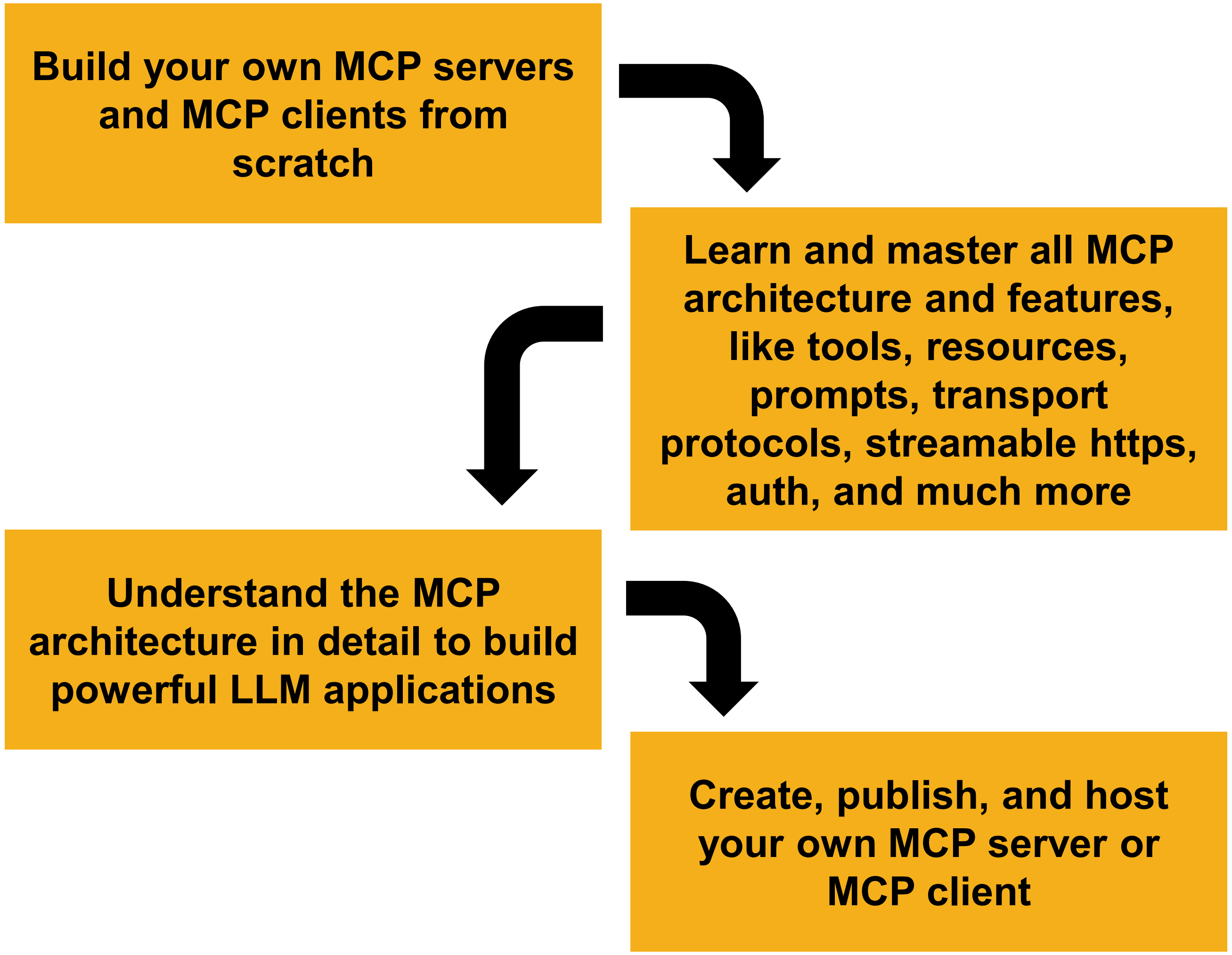


```
{
  "mcpServers": {
    "airbnb": {
      "command": "npx",
      "args": [
        "-y",
        "@openbnb/mcp-server-airbnb"
      ]
    }
  }
}
```

© Henry Habib

What is this course?

MCP Masterclass



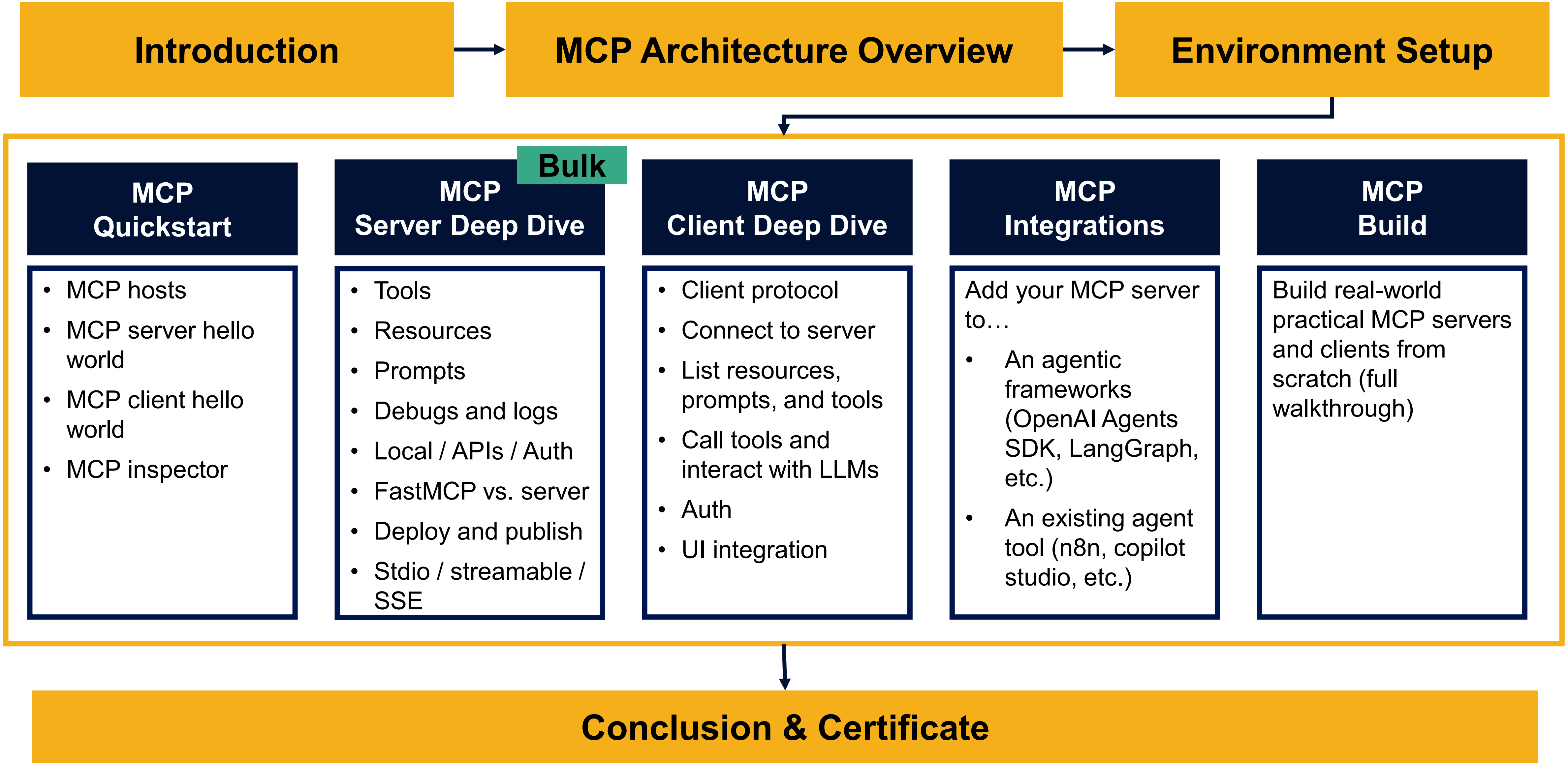
Build your own MCP servers and MCP clients from scratch

Learn and master all MCP architecture and features, like tools, resources, prompts, transport protocols, streamable https, auth, and much more

Understand the MCP architecture in detail to build powerful LLM applications

Create, publish, and host your own MCP server or MCP client *https://linktr.ee/henrylearning*

Course Roadmap



Introduction → MCP Architecture Overview → Environment Setup

MCP Quickstart

- MCP hosts
- MCP server hello world
- MCP client hello world
- MCP inspector

MCP Server Deep Dive Bulk

- Tools
- Resources
- Prompts
- Debugs and logs
- Local / APIs / Auth
- FastMCP vs. server
- Deploy and publish
- Stdio / streamable / SSE

MCP Client Deep Dive

- Client protocol
- Connect to server
- List resources, prompts, and tools
- Call tools and interact with LLMs
- Auth
- UI integration

MCP Integrations

Add your MCP server to…

- An agentic frameworks (OpenAI Agents SDK, LangGraph, etc.)
- An existing agent tool (n8n, copilot studio, etc.)

MCP Build

Build real-world practical MCP servers and clients from scratch (full walkthrough)

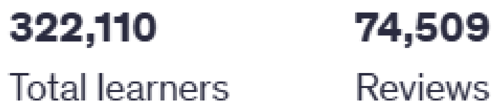
Conclusion & Certificate

Who Am I?

Automation / Productivity Consultant

Productivity / Gen AI / No-Code

Instructor



322,110
Total learners

74,509
Reviews

Please rate this course, leave feedback, ask questions, message me, and have fun!

Keys To Success

Do, don’t watch

Explore

Ask questions / get involved!

Ways to reach out and contact me

Link Tree

Direct Message

Udemy Q&A

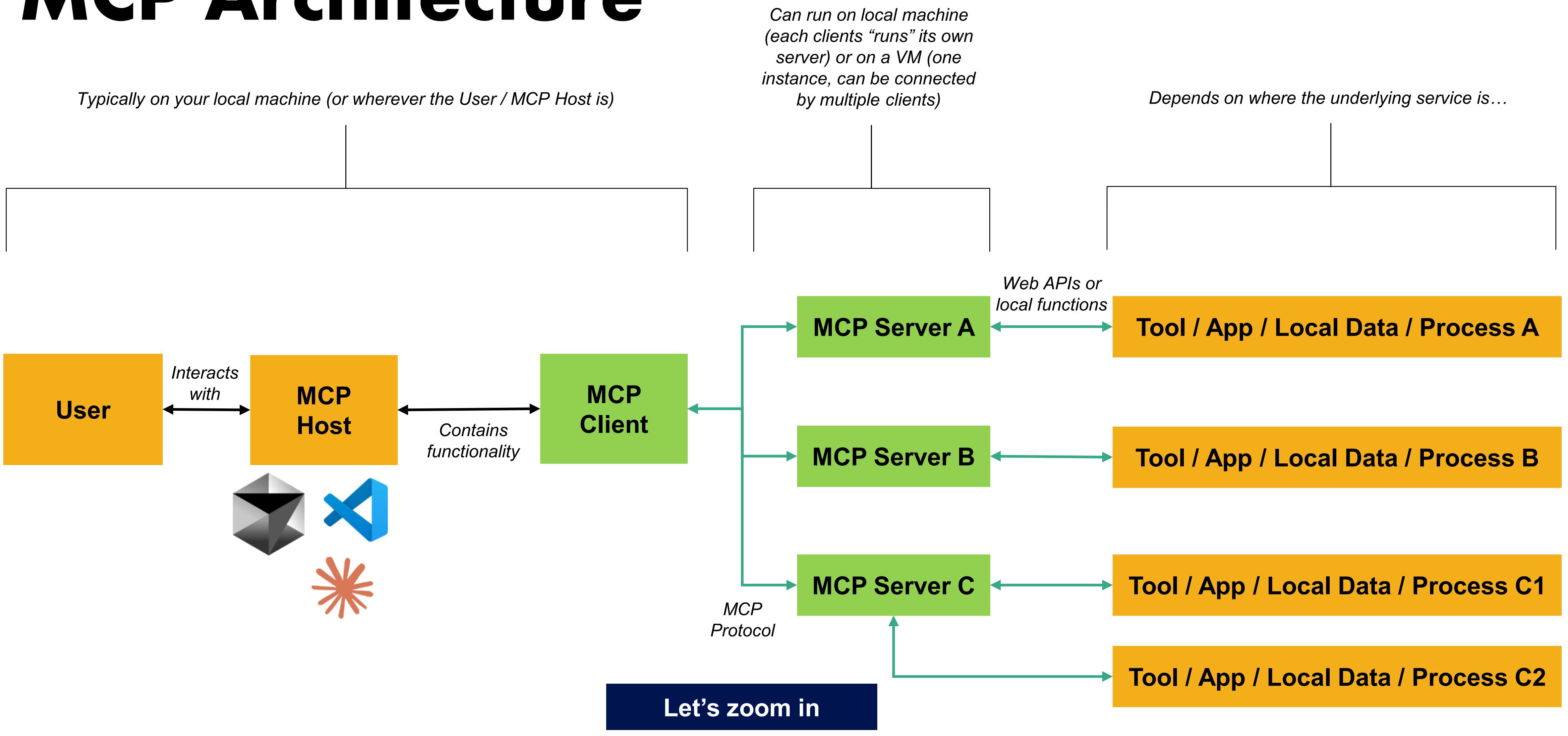
See linktr.ee/henrylearning

Leave a Rating

Please rate this course, leave feedback, ask questions, message me, and have fun!

MCP Architecture

https://linktr.ee/henrylearning



Typically on your local machine (or wherever the User / MCP Host is)

Can run on local machine (each clients "runs" its own server) or on a VM (one instance, can be connected by multiple clients)

Depends on where the underlying service is…

User

Interacts with

MCP Host

Contains functionality

MCP Client

MCP Protocol

MCP Server A

MCP Server B

MCP Server C

Web APIs or local functions

Tool / App / Local Data / Process A

Tool / App / Local Data / Process B

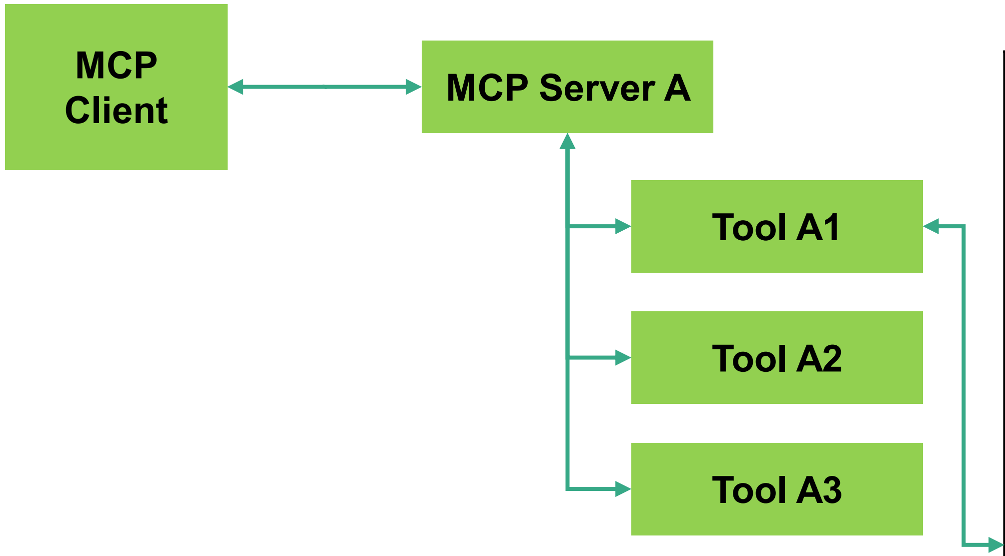
Tool / App / Local Data / Process C1

Tool / App / Local Data / Process C2

Let's zoom in

https://linktr.ee/henrylearning

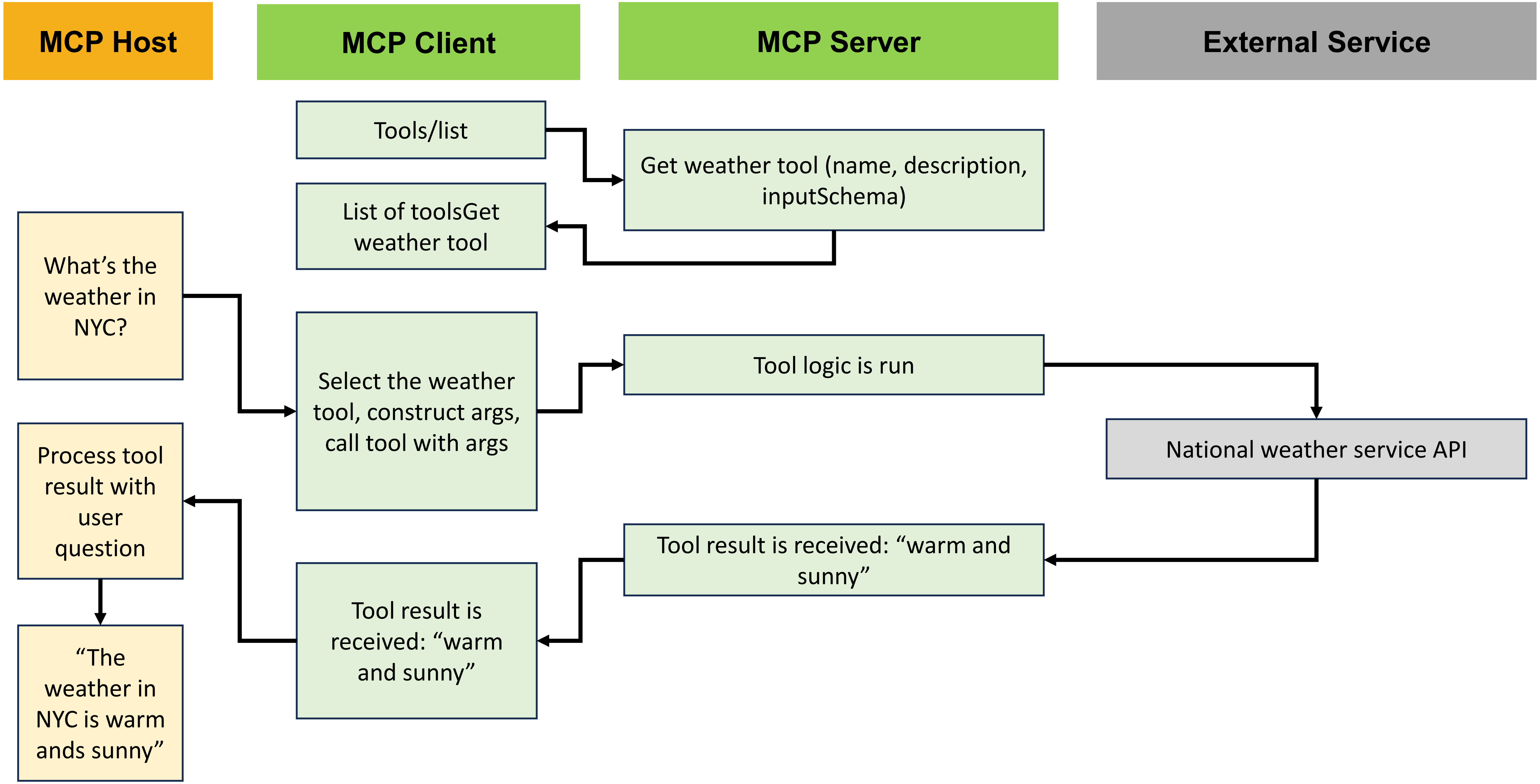
MCP Server Deep Dive



Parameter	Purpose	Example
Name	Name of the tool or function	“get weather”
Description	Description of what the tool or function does, along with what arguments are expected	“gets the weather given a location. Location should be a string”
Input Schema	Dictionary of arguments that the tool or function accepts	{“location”: “string”}

https://linktr.ee/henrylearning

MCP Client - Server Communication



MCP Host

MCP Client

MCP Server

External Service

Tools/list

Get weather tool (name, description, inputSchema)

List of toolsGet weather tool

What's the weather in NYC?

Select the weather tool, construct args, call tool with args

Tool logic is run

National weather service API

Tool result is received: "warm and sunny"

Process tool result with user question

Tool result is received: "warm and sunny"

"The weather in NYC is warm ands sunny"

https://linktr.ee/henrylearning

MCP Server Primitives

Tools	Resources	Prompts
Model controlled logic / functions that can be invoked and does something	*Application controlled data to provide contextual data to the host / client*	*User-controlled prompt templates to provide LLMs with custom prompts*
API requests, CRUD operations, computations, etc.	*File contents, read instructions, user data, etc.*	*Prompts to craft research report in a specific way*

https://linktr.ee/henrylearning

Server vs. FastMCP

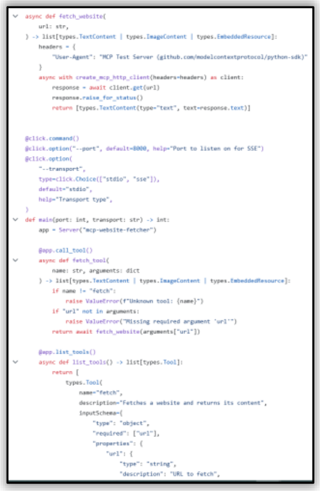
Server

Original MCP server implementation

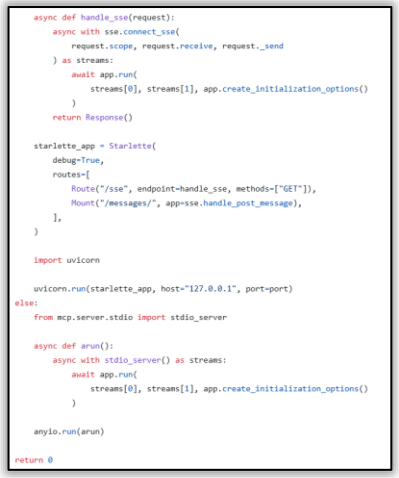
↓

FastMCP

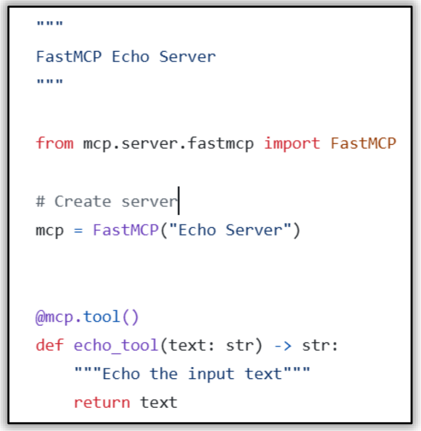
Wrapper to make things simpler and easier



```python
async def fetch_website(
    url: str,
) -> list[types.TextContent | types.ImageContent | types.EmbeddedResource]:
    headers = {
        "User-Agent": "MCP Test Server (github.com/modelcontextprotocol/python-sdk)"
    }
    async with create_mcp_http_client(headers=headers) as client:
        response = await client.get(url)
        response.raise_for_status()
        return [types.TextContent(type="text", text=response.text)]


@click.command()
@click.option("--port", default=8000, help="Port to listen on for SSE")
@click.option(
    "--transport",
    type=click.Choice(["stdio", "sse"]),
    default="stdio",
    help="Transport type",
)
def main(port: int, transport: str) -> int:
    app = Server("mcp-website-fetcher")

    @app.call_tool()
    async def fetch_tool(
        name: str, arguments: dict
    ) -> list[types.TextContent | types.ImageContent | types.EmbeddedResource]:
        if name != "fetch":
            raise ValueError(f"Unknown tool: {name}")
        if "url" not in arguments:
            raise ValueError("Missing required argument 'url'")
        return await fetch_website(arguments["url"])

    @app.list_tools()
    async def list_tools() -> list[types.Tool]:
        return [
            types.Tool(
                name="fetch",
                description="Fetches a website and returns its content",
                inputSchema={
                    "type": "object",
                    "required": ["url"],
                    "properties": {
                        "url": {
                            "type": "string",
                            "description": "URL to fetch",
```



```python
    async def handle_sse(request):
        async with sse.connect_sse(
            request.scope, request.receive, request._send
        ) as streams:
            await app.run(
                streams[0], streams[1], app.create_initialization_options()
            )
        return Response()

    starlette_app = Starlette(
        debug=True,
        routes=[
            Route("/sse", endpoint=handle_sse, methods=["GET"]),
            Mount("/messages/", app=sse.handle_post_message),
        ],
    )

    import uvicorn

    uvicorn.run(starlette_app, host="127.0.0.1", port=port)
else:
    from mcp.server.stdio import stdio_server

    async def arun():
        async with stdio_server() as streams:
            await app.run(
                streams[0], streams[1], app.create_initialization_options()
            )

    anyio.run(arun)

return 0
```



```python
"""
FastMCP Echo Server
"""

from mcp.server.fastmcp import FastMCP

# Create server
mcp = FastMCP("Echo Server")


@mcp.tool()
def echo_tool(text: str) -> str:
    """Echo the input text"""
    return text
```

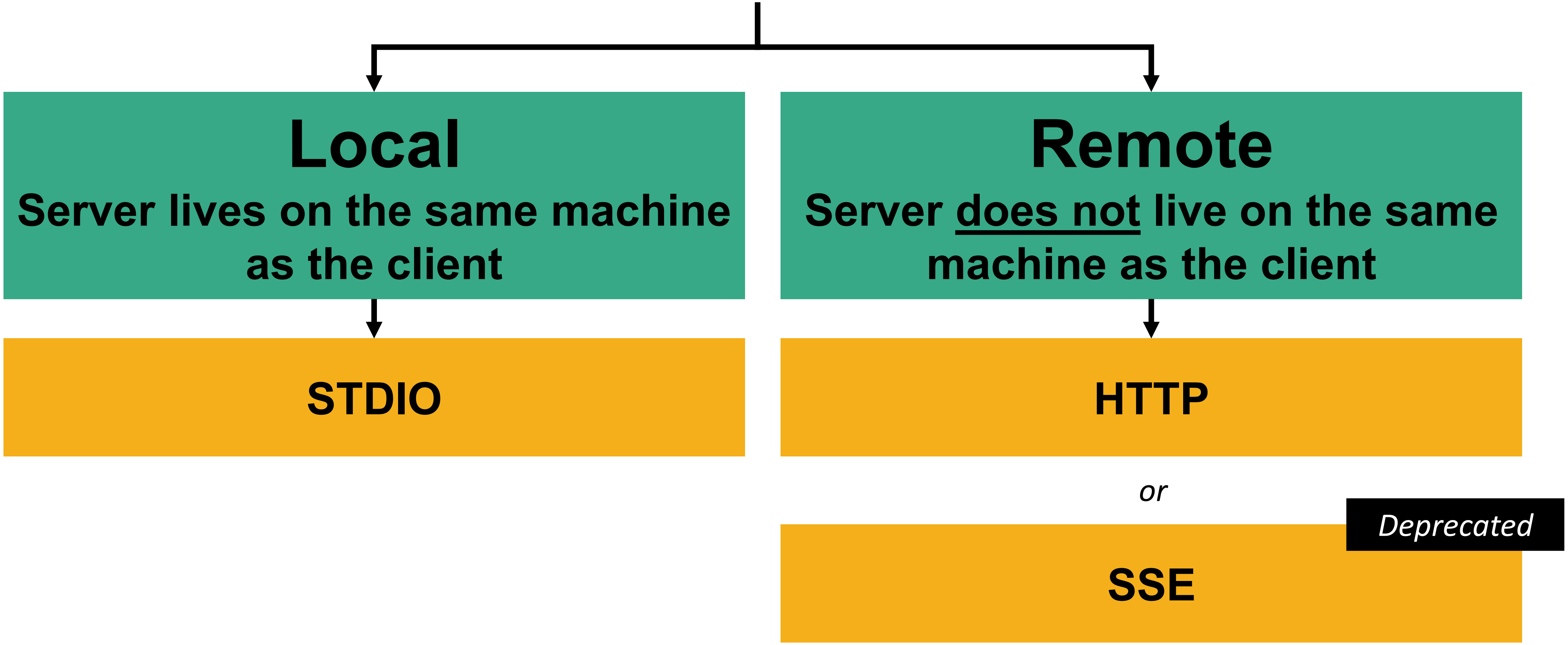
Always use FastMCP wherever possible – it's an abstraction that support most, if not all, features and is heavily supported by the community and the open-source project

MCP Transport Mechanisms

https://linktr.ee/henrylearning

Local vs. Remote

Where is the server located?



Local

Server lives on the same machine as the client

STDIO

Remote

Server <u>does not</u> live on the same machine as the client

HTTP

or

SSE

Deprecated

© Henry Habib

MCP Transport Mechanisms
Local vs. Remote

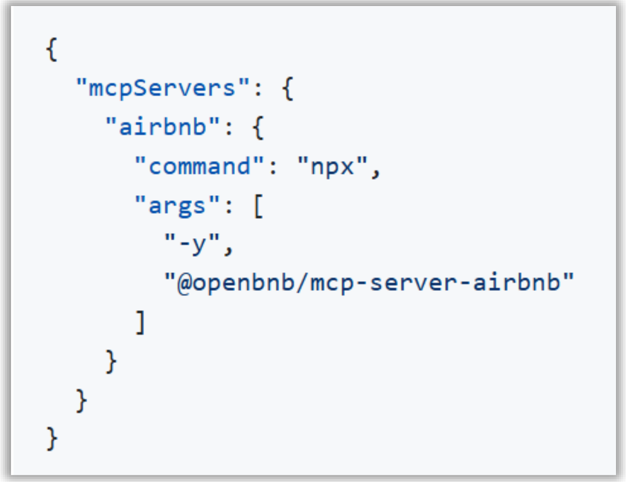
https://linktr.ee/henrylearning

Local (STDIO)

- *Uses the STDIO transport mechanism*
- *Server is run on your local machine (the same place that your MCP host / client is run)*
- *There is always one server running for each* MCP client (each user runs their own MCP server)
- *Benefits*
    - *Necessary for tools that contain local processes (i.e., affect / take actions on your local machine)*
    - *Simple setup and installation, auth less necessary as both client and server run on same machine*

Process

- *User tells MCP Host / Client on how to download and run the server, typically through a config script like the one below*
- *MCP Host / Client downloads the MCP server on local machine, installs it, and runs it as a sub-process… MCP server is running locally*



```
{
  "mcpServers": {
    "airbnb": {
      "command": "npx",
      "args": [
        "-y",
        "@openbnb/mcp-server-airbnb"
      ]
    }
  }
}
```

© Henry Habib

MCP Transport Mechanisms
Local vs. Remote

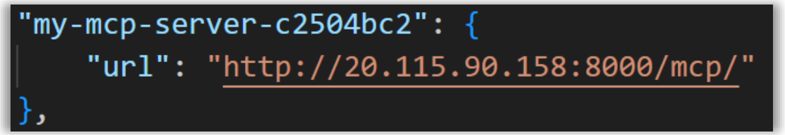
https://linktr.ee/henrylearning

Remote (HTTP or SSE)

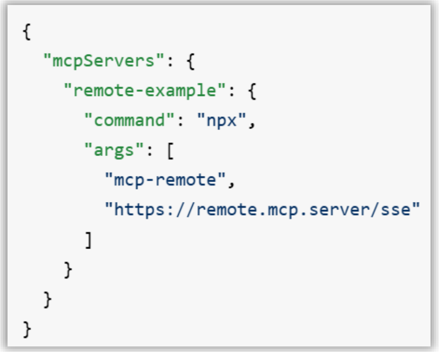
- *Uses the **Streamable HTTP** transport mechanism*
- *Server is run on a virtual machine (NOT the same place that your MCP host / client is run)*
- *There is one server running* for all MCP clients that connect to it
- Any logic is run on the virtual machine service
- *Benefits*
  - *Latest and greatest server is always available to all clients*
  - *Processing / logic does not happen on local machine*
  - *Portability, even easier to install*
  - *Works with online MCP clients / hosts*

Process

- *User tells MCP Host / Client on where the MCP server runs*
- *MCP Host / Client pings the MCP server through HTTP*



```
"my-mcp-server-c2504bc2": {
    "url": "http://20.115.90.158:8000/mcp/"
},
```



```
{
  "mcpServers": {
    "remote-example": {
      "command": "npx",
      "args": [
        "mcp-remote",
        "https://remote.mcp.server/sse"
      ]
    }
  }
}
```

© Henry Habib

Congratulations

Leave a Rating

Please rate this course, leave feedback, ask questions, message me, and have fun!

Certificate