

# Create AI Agents with Model Context Protocol (MCP)

Hands-on intro to AI Agents using the MCP standard

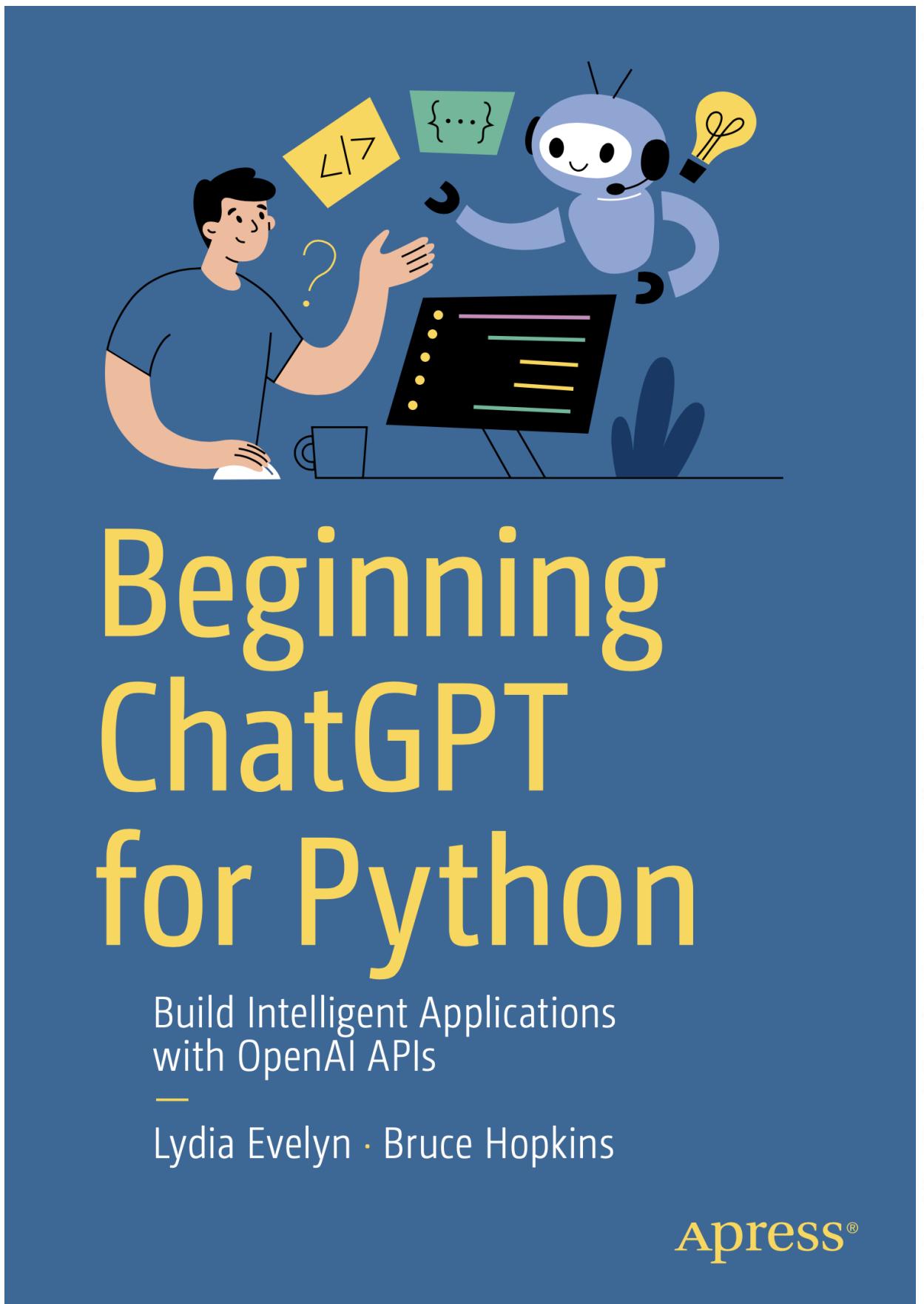
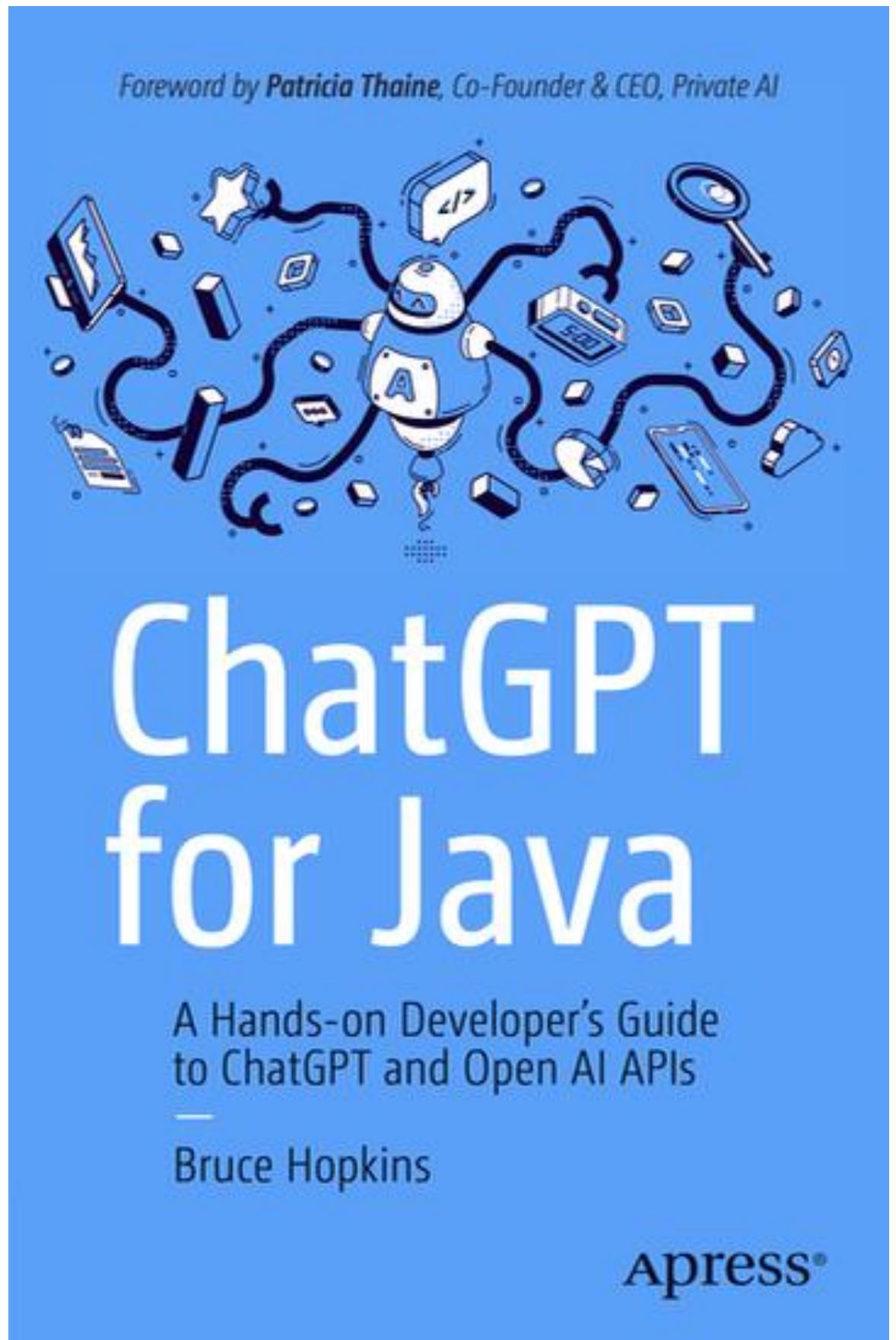


**Bruce Hopkins**

Author, “ChatGPT for Java”,  
2024

# About the presenter

- Intel Software Innovator for AI
- Oracle Java Champion
- Book Author



# Audience Poll Question #1



# Audience Poll Question #1

**Which category best suits your industry?** (Single response)

- Technology and IT
- Manufacturing
- E-commerce and retail
- Finance
- Government
- Higher education
- Personal/other



Focus

# Official MCP SDKs

## Official MCP SDKs



# Audience Poll Question #2

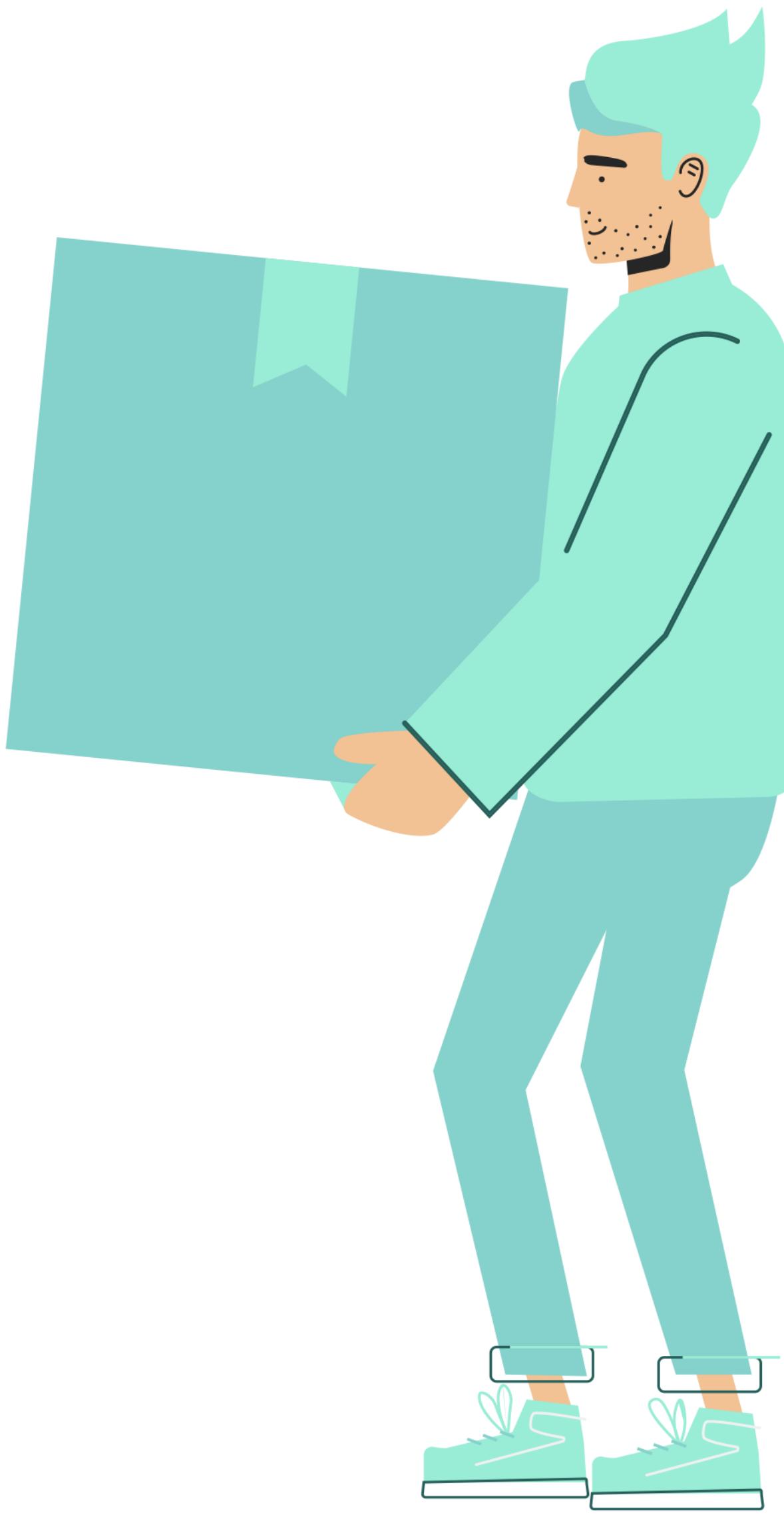


# Audience Poll Question #2

**Have You Created AI Agents Before? (Single response)**

- Yes, using Open AI APIs
- Yes, using Anthropic APIs
- Yes, using other APIs
- No

# What are the Key Takeaways from This Course?



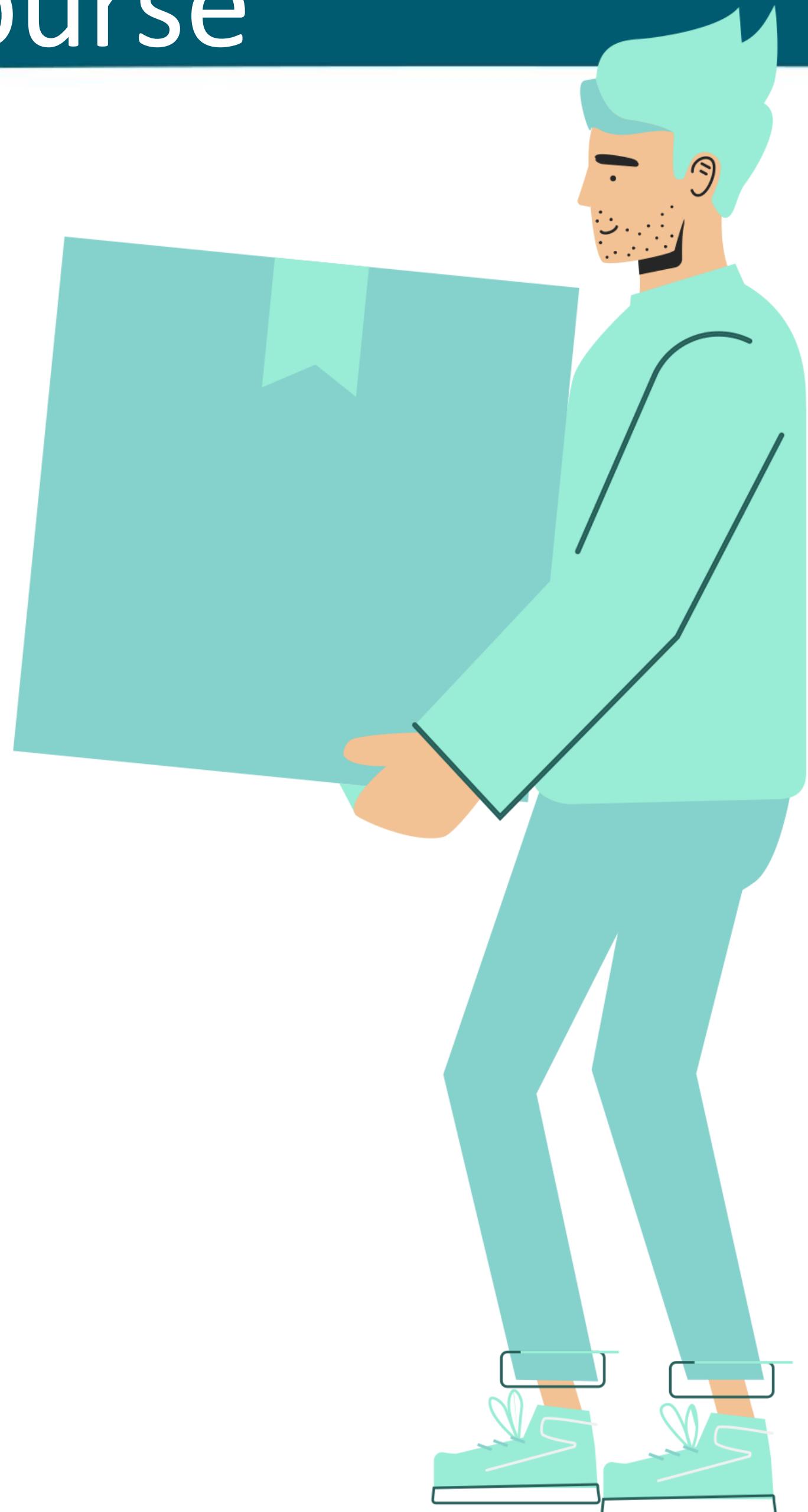
# Key Takeaways from This Course

## PART 1

You will **understand the basics** of the Model Context Protocol (MCP)

Understand the problems that it solves

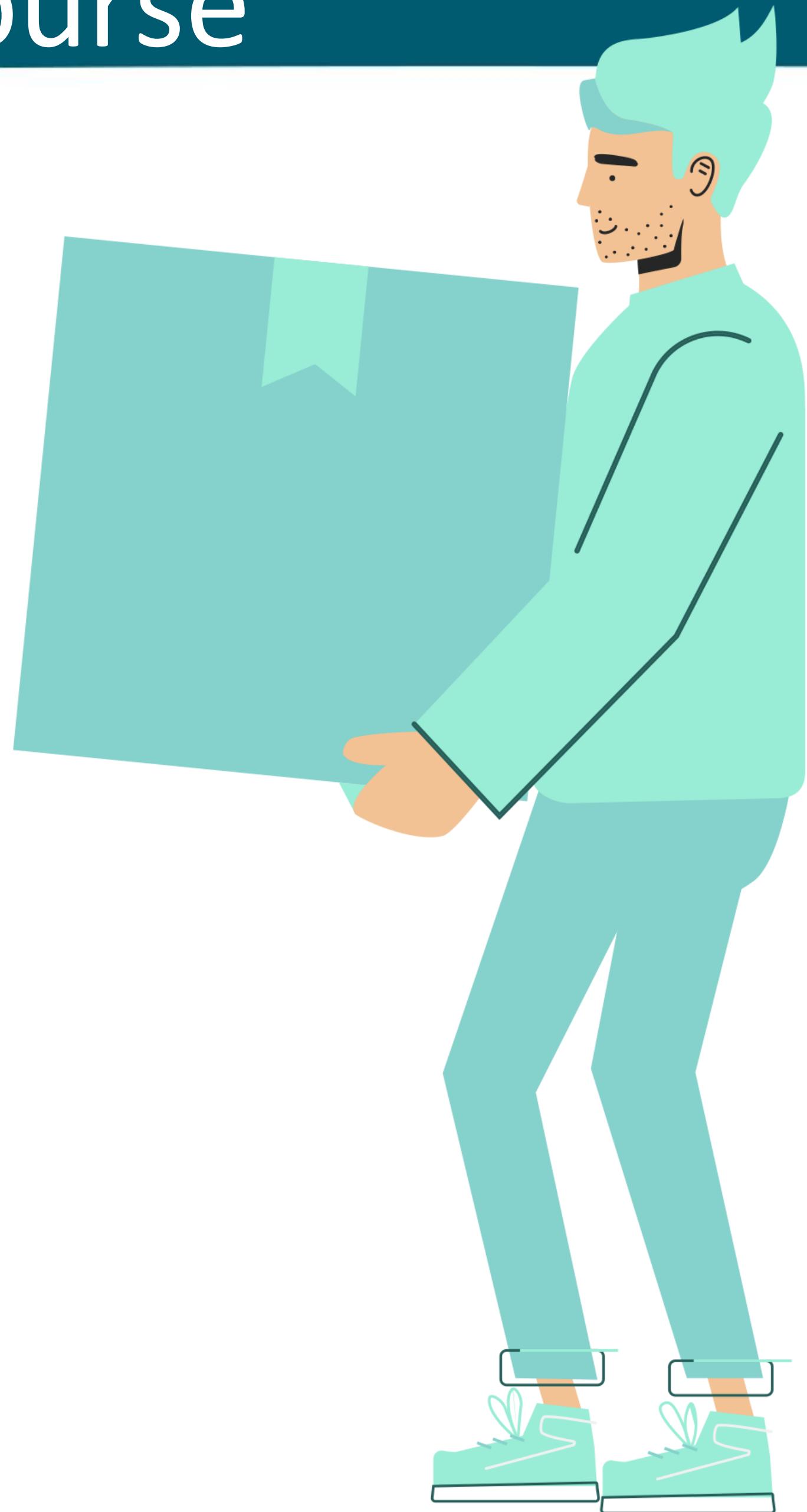
You'll understand why the industry has given overwhelming support



# Key Takeaways from This Course

## PART 2

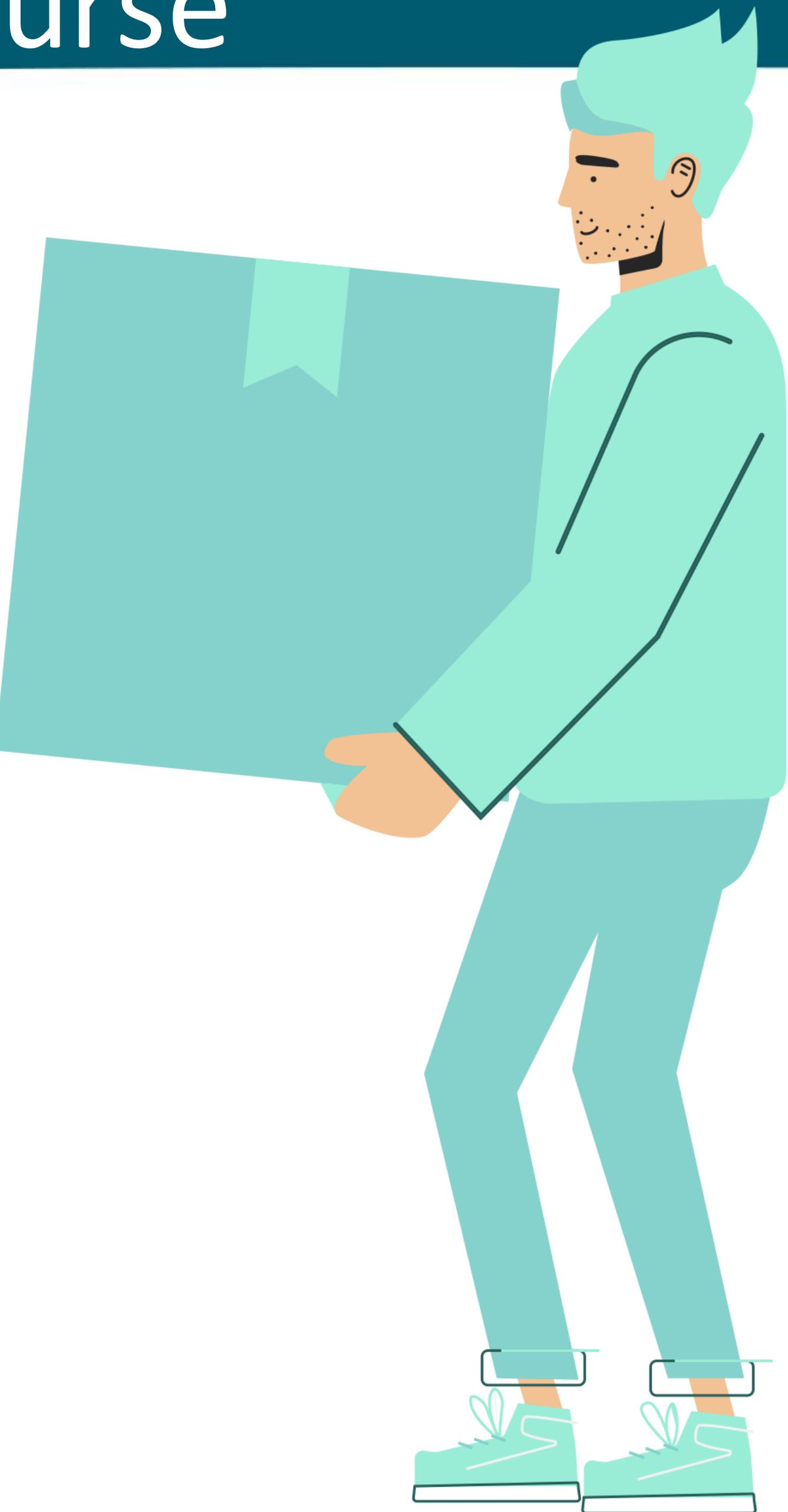
You take a deep dive into the MCP Protocol and learn the ins-and-outs of **how MCP Servers work**



# Key Takeaways from This Course

## PART 3

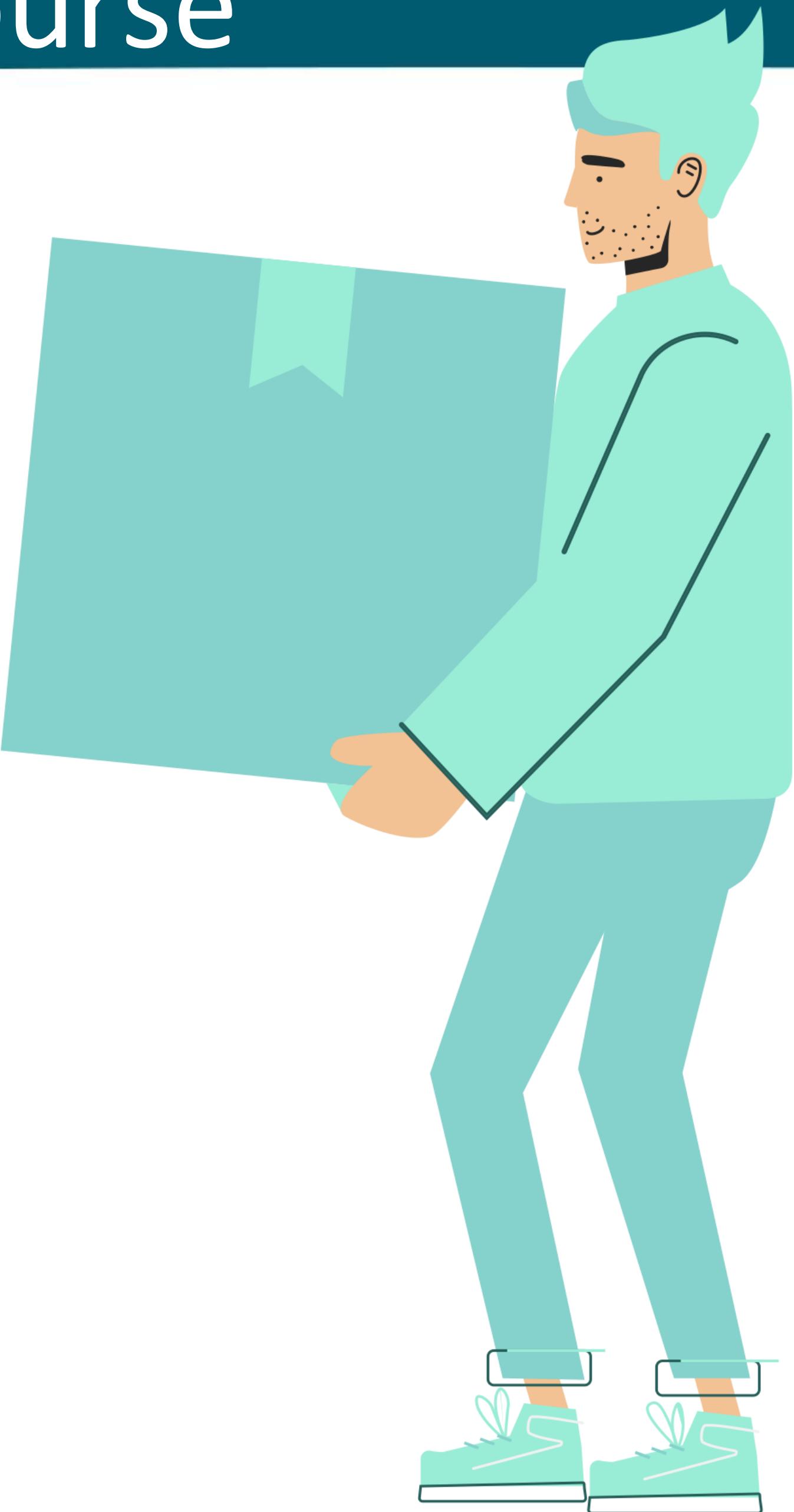
You'll learn hands-on how to build  
an MCP Server in **Python**



# Key Takeaways from This Course

## PART 4

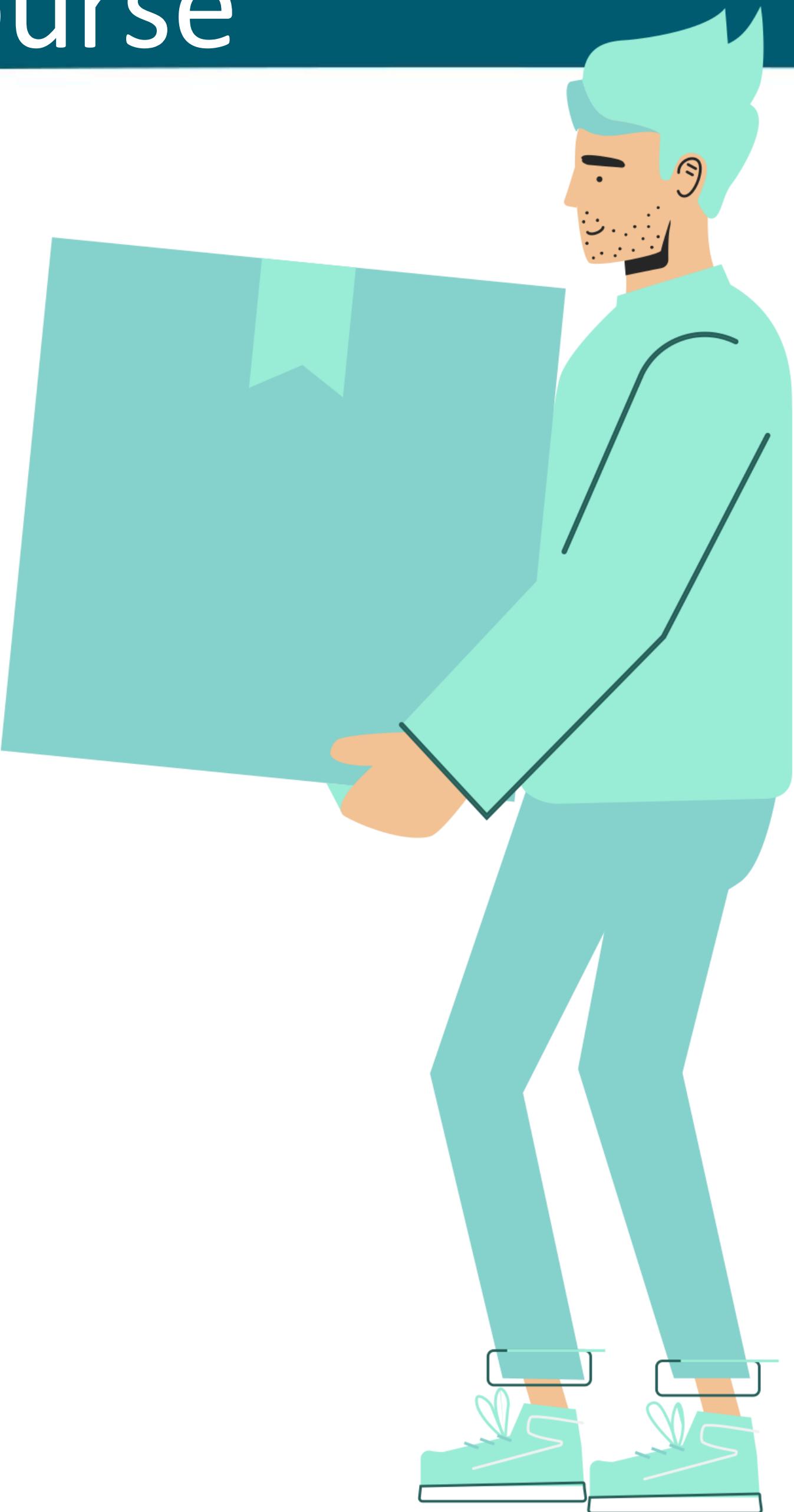
You'll learn hands-on how to build  
an MCP Server in **Java**



# Key Takeaways from This Course

## PART 5

You'll learn hands-on how to build  
an MCP Server in **Node.js**

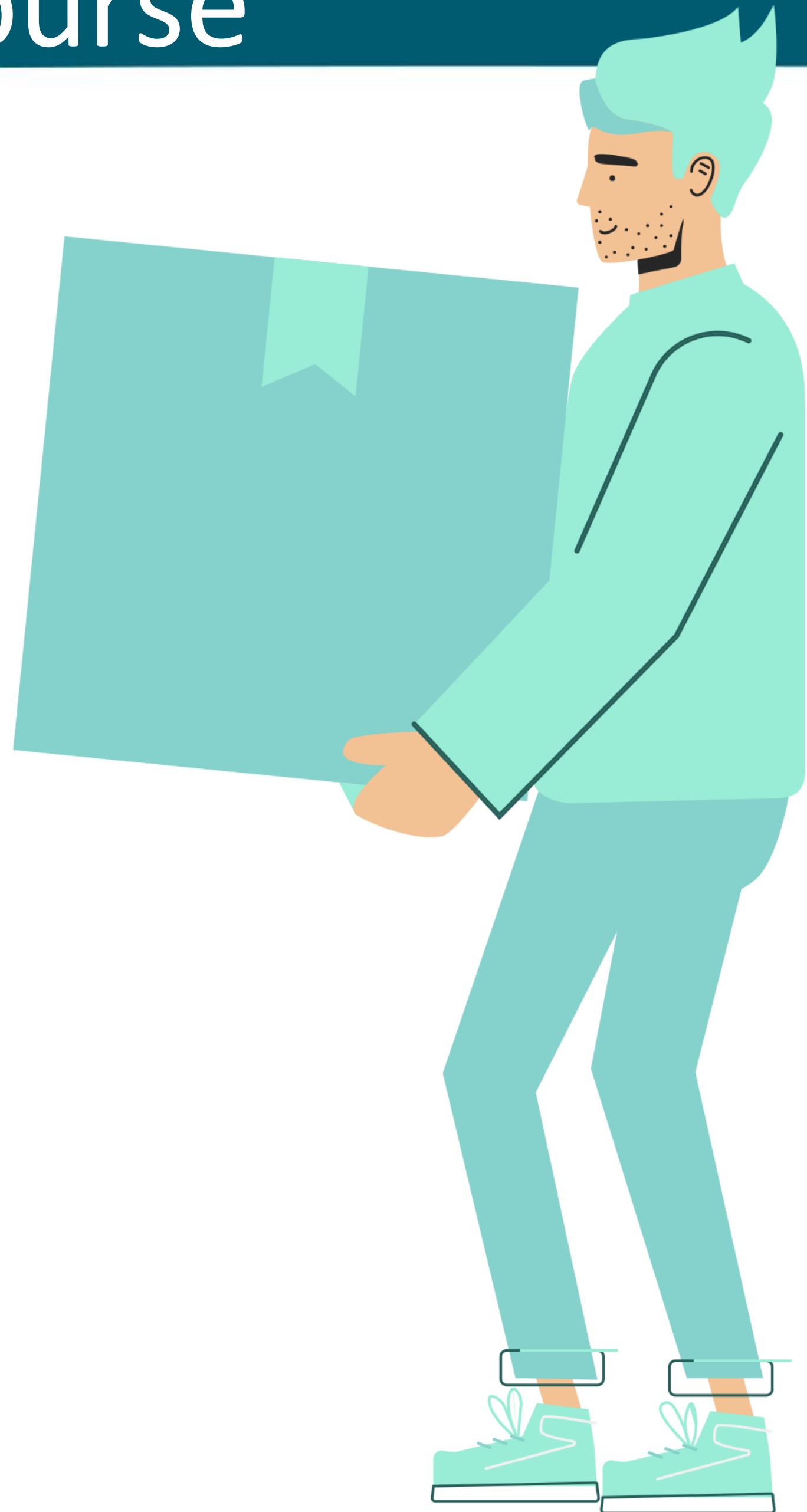


# Key Takeaways from This Course

## PART 6

We will switch focus from building MCP Servers from scratch to **leveraging existing** MCP Servers

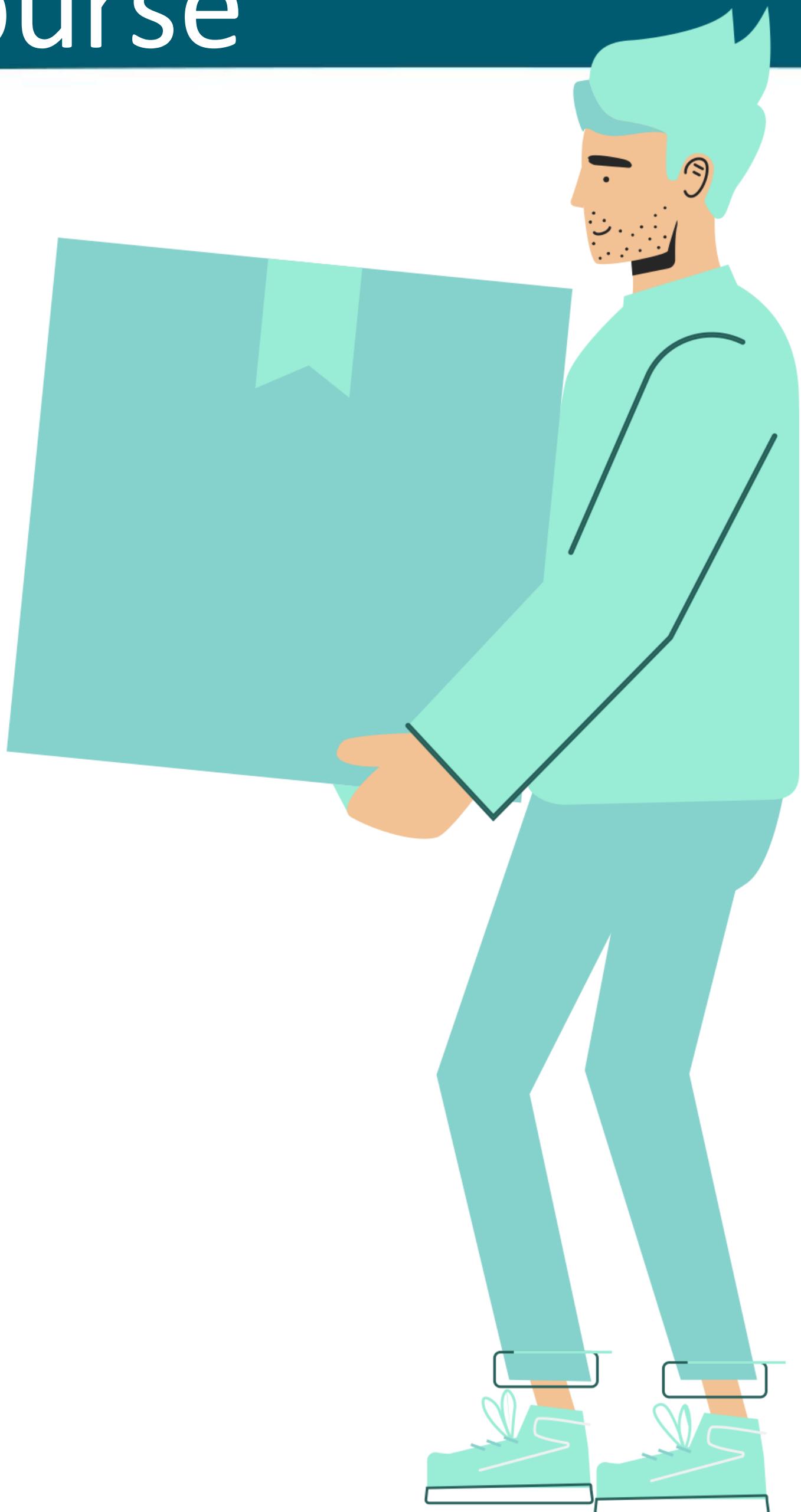
You will learn how to use and configure the **MCP Server for PostgreSQL**



# Key Takeaways from This Course

## Course Wrapup

I'll provide my personal contact information for after course followup



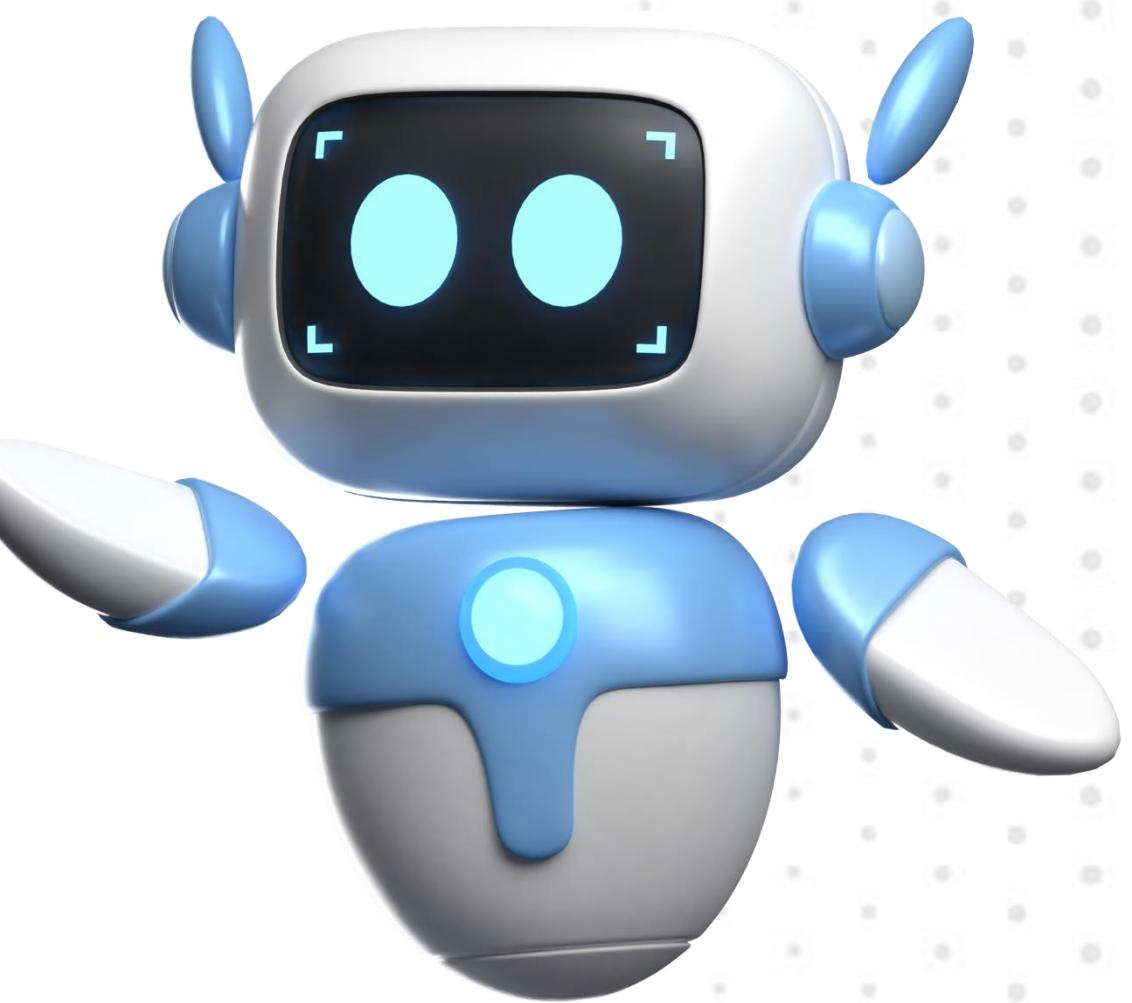
# PART 1

Introducing the Model Context  
Protocol (MCP) for AI Agents:  
**MCP Clients**



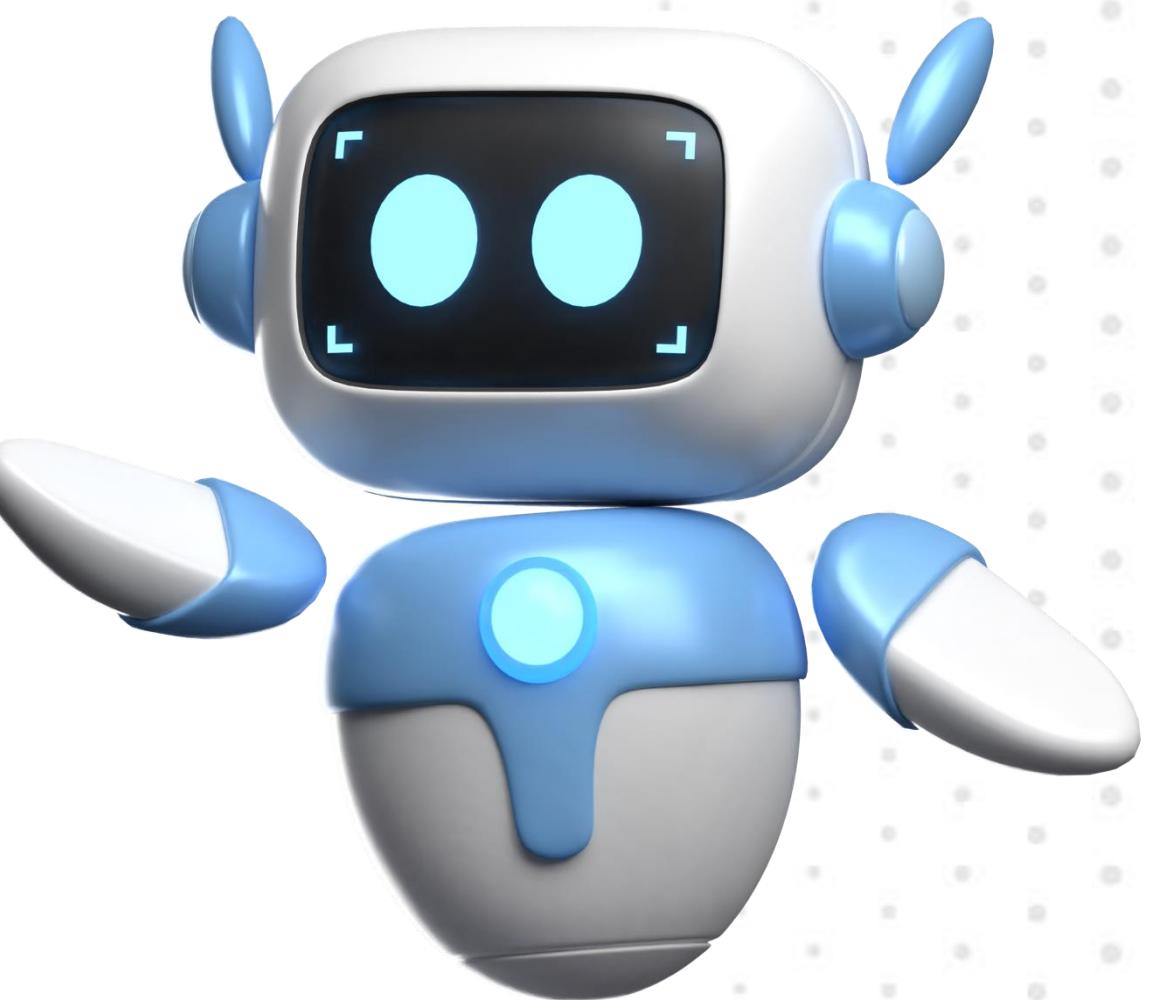
# Why do we need AI agents?

- It's common knowledge why we need AI agents



# Why do we need AI agents?

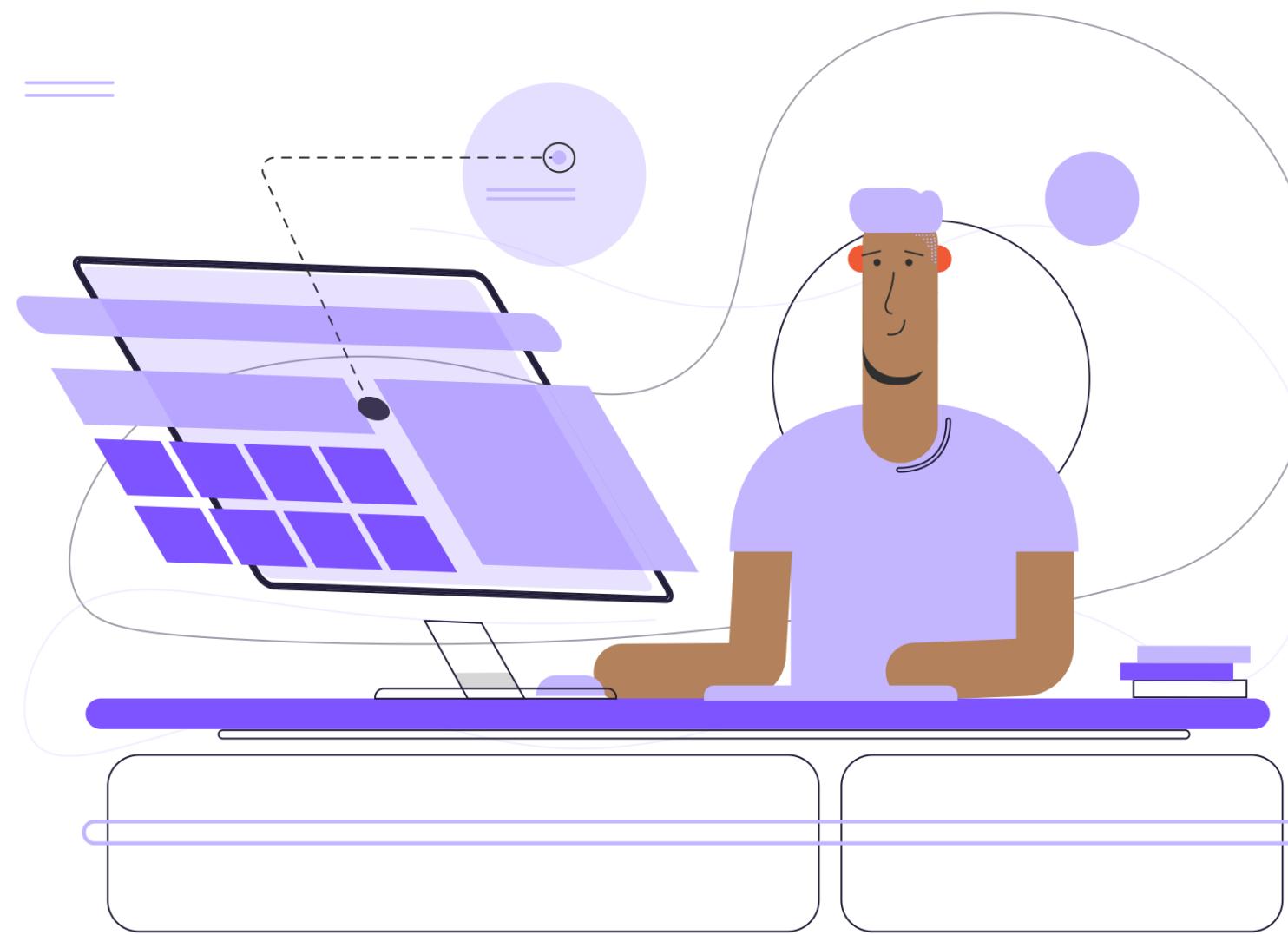
- It's common knowledge why we need AI agents
- It's all about the **3 domains** of information



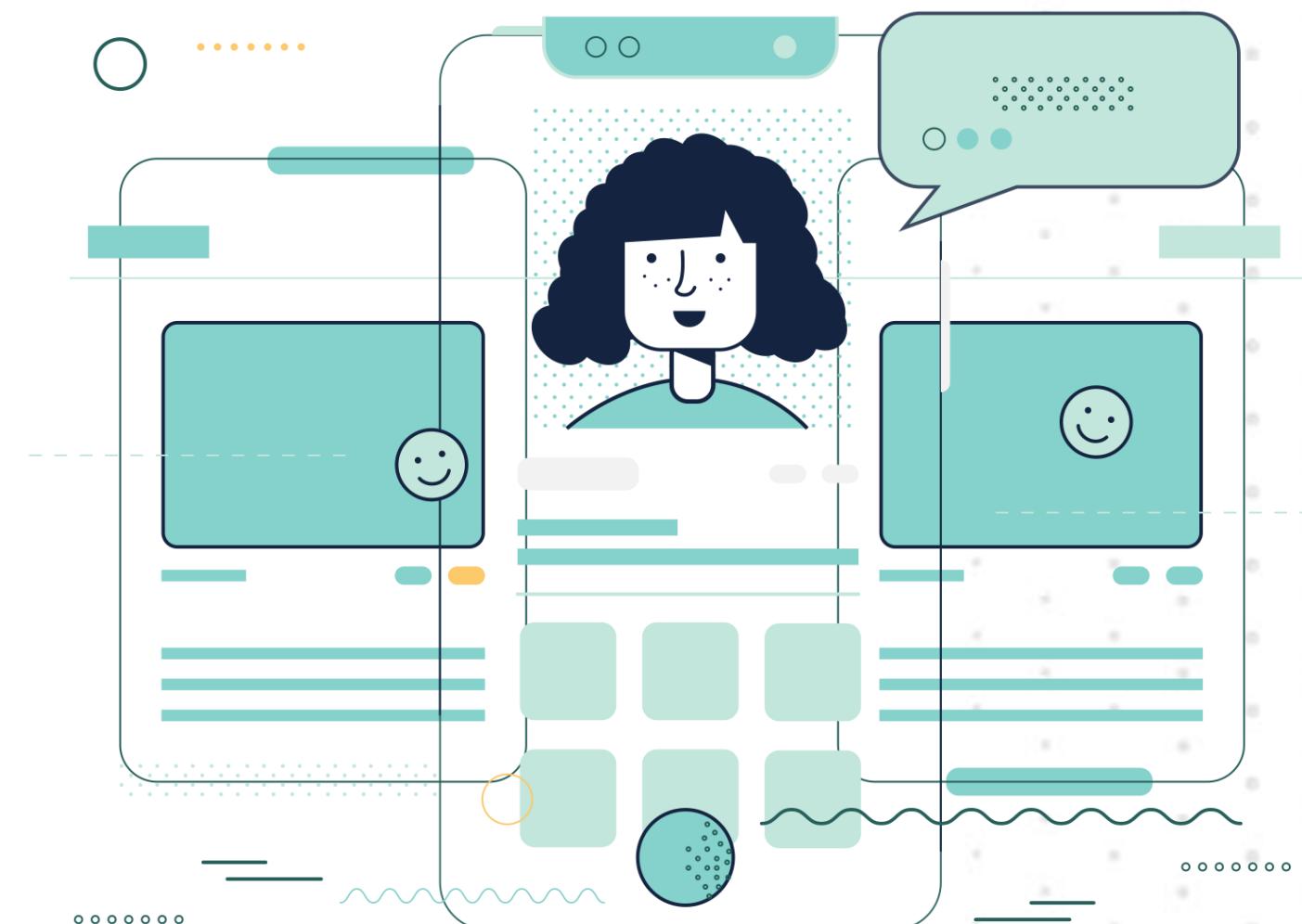
# There are Basically 3 Domains of Information



**Internet**



**Workplace**



**You/Home**

# The Internet domain

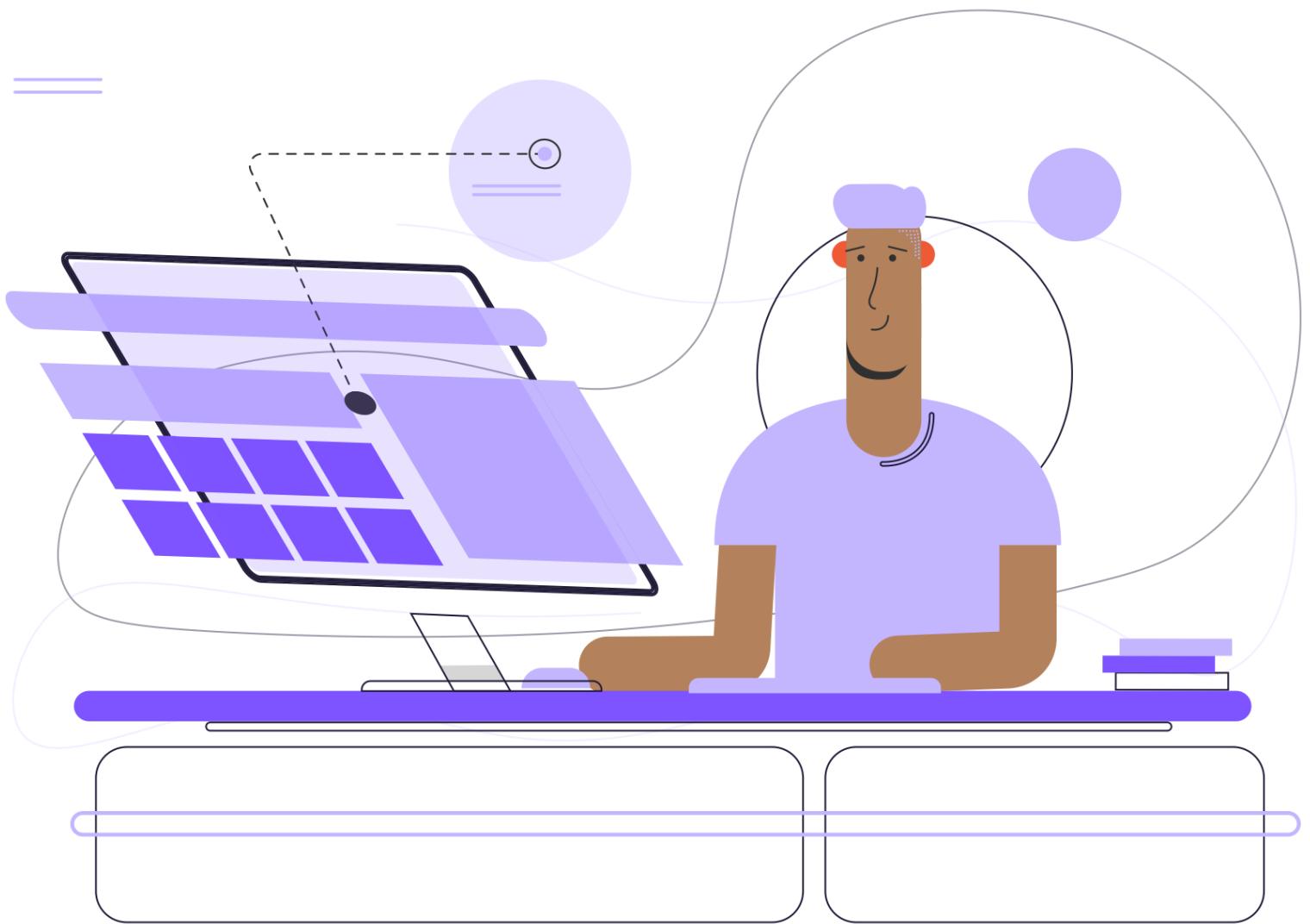
- ChatGPT is trained on most (but not all) of this information



Internet

# The workplace domain

- ChatGPT is **NOT** trained on the information available within your company

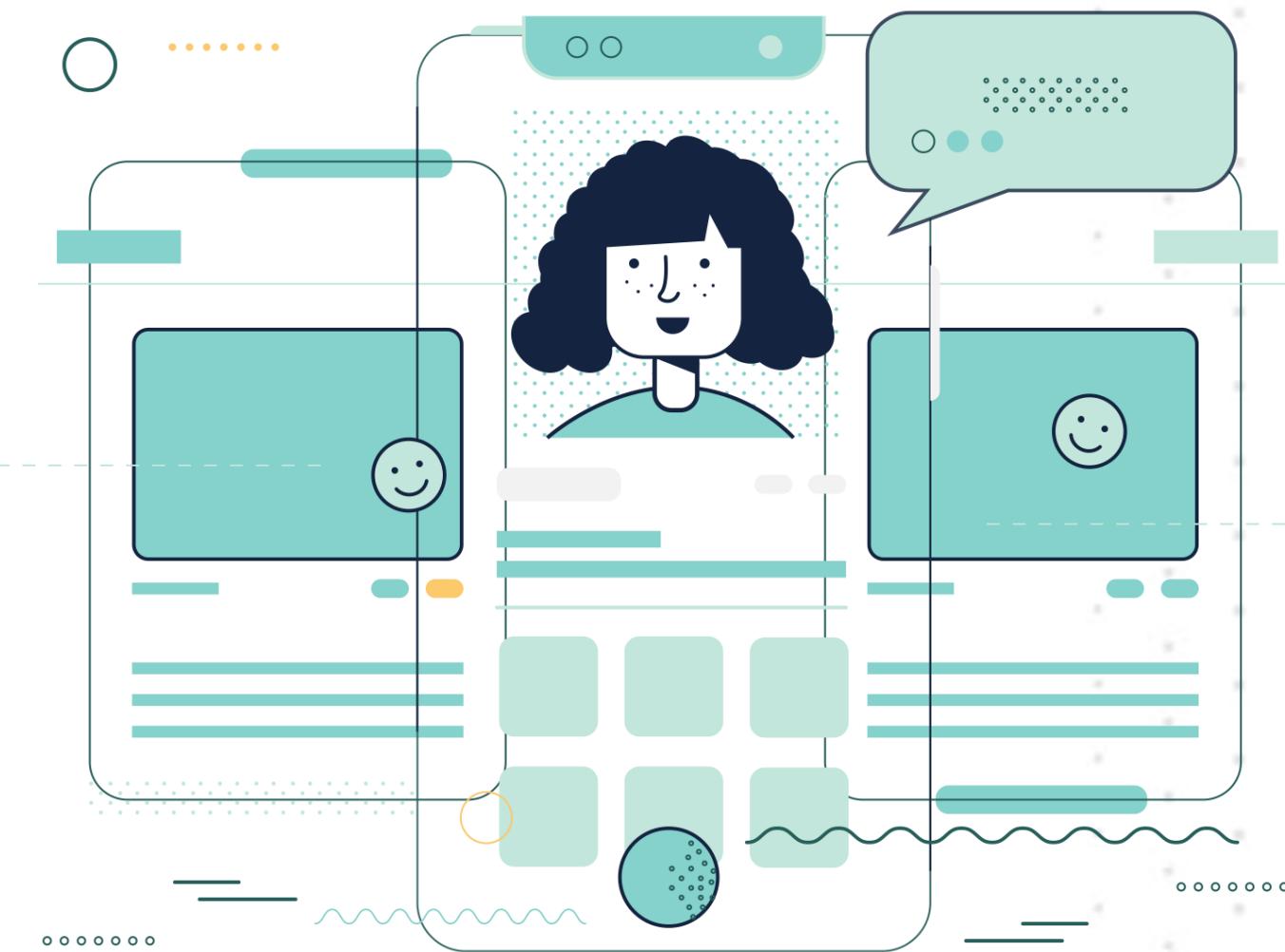


- This information is secured behind firewalls and data silos

**Workplace**

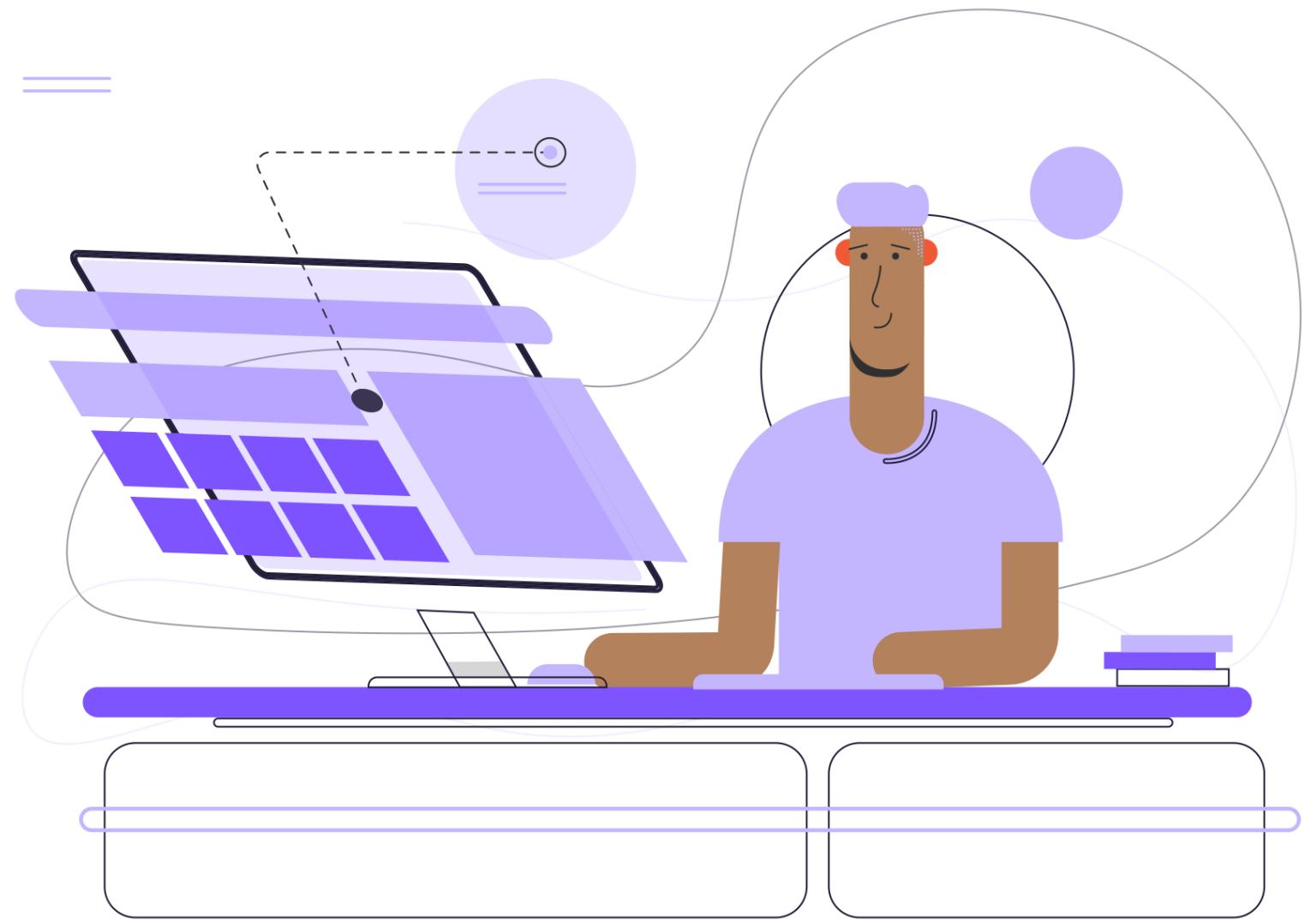
# The personal/home domain

- ChatGPT is **NOT** trained on your personal/private the information
- This information is secured on your phone or PC

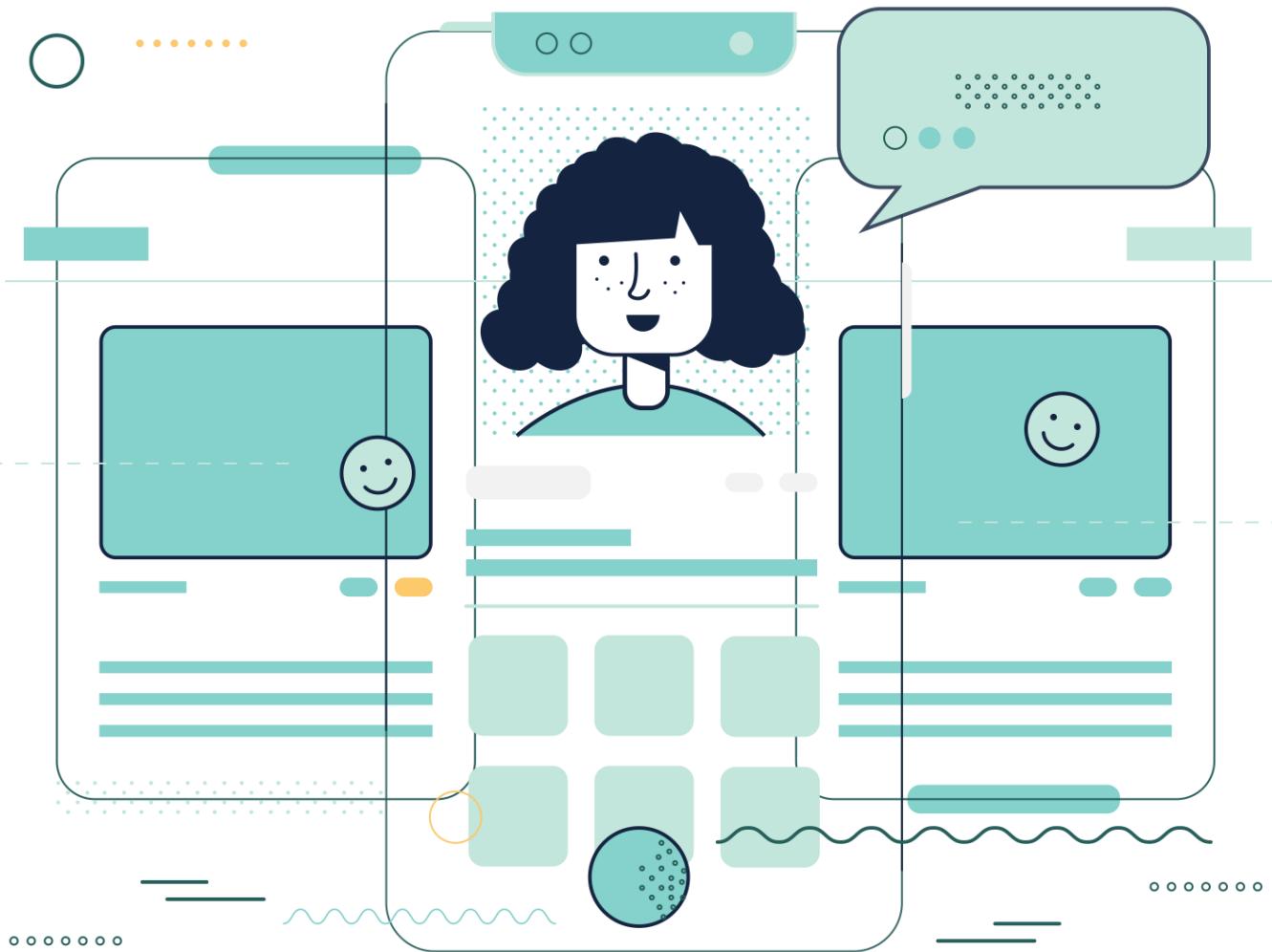


You/Home

# Why we need AI Agents



**Workplace**



**You/Home**

# Can ChatGPT Get the Weather?

## DEMO 1

# Can ChatGPT Get the Weather?

## DEMO 1



### ChatGPT Playground

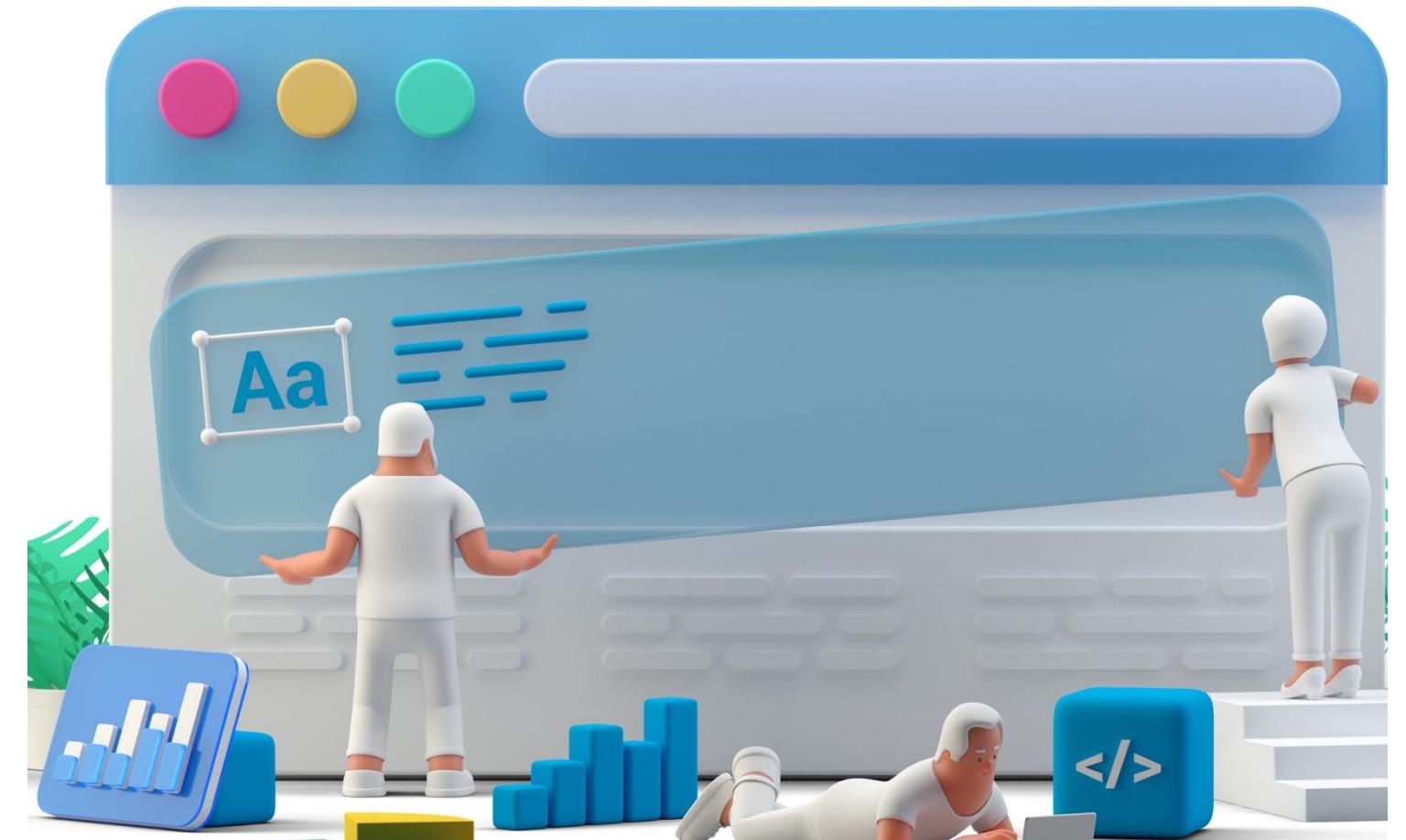
# Can ChatGPT Get the Weather?

## DEMO 1



### PROMPT:

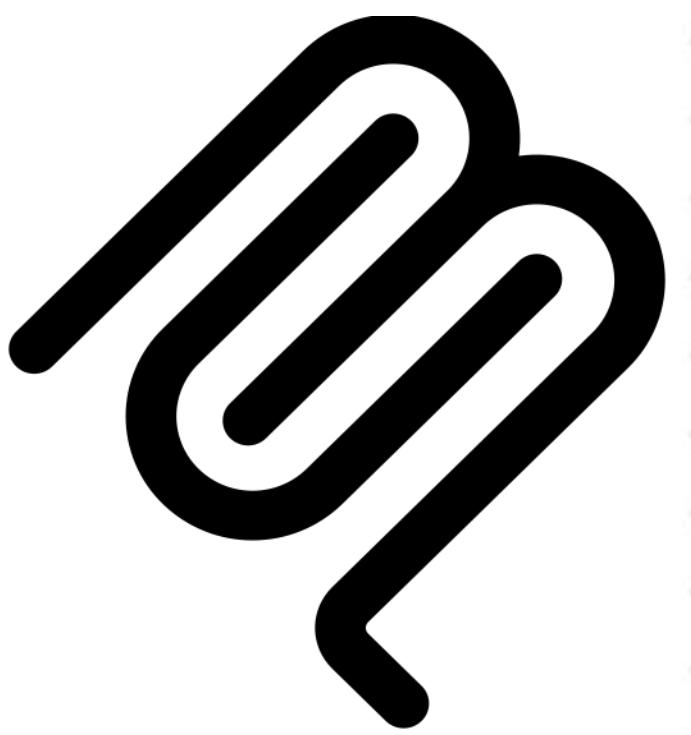
*What's the weather in Dallas?*



ChatGPT Playground

# Why was MCP Created?

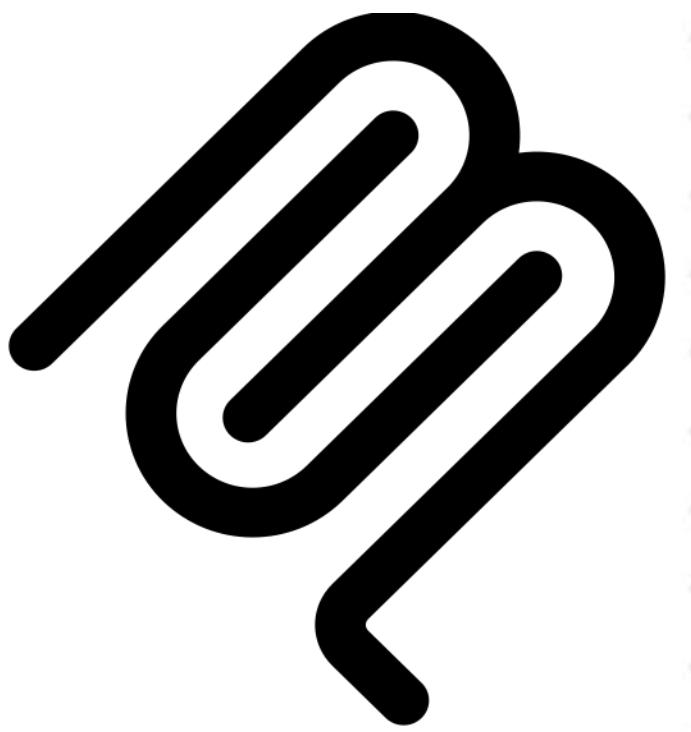
- There were no standards



**Model Context Protocol**

# Why was MCP Created?

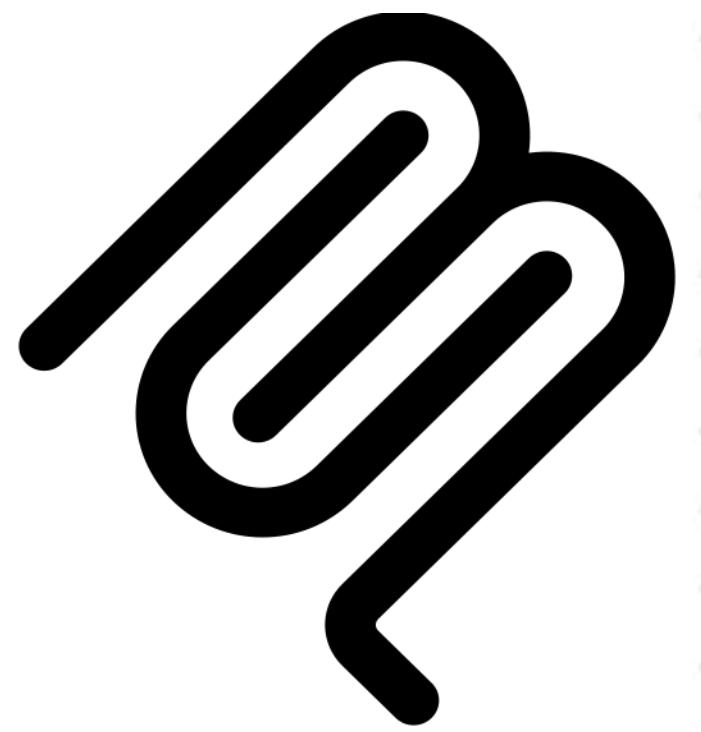
- There were no standards
- It's like the Wild Wild West,  
but for AI



**Model Context Protocol**

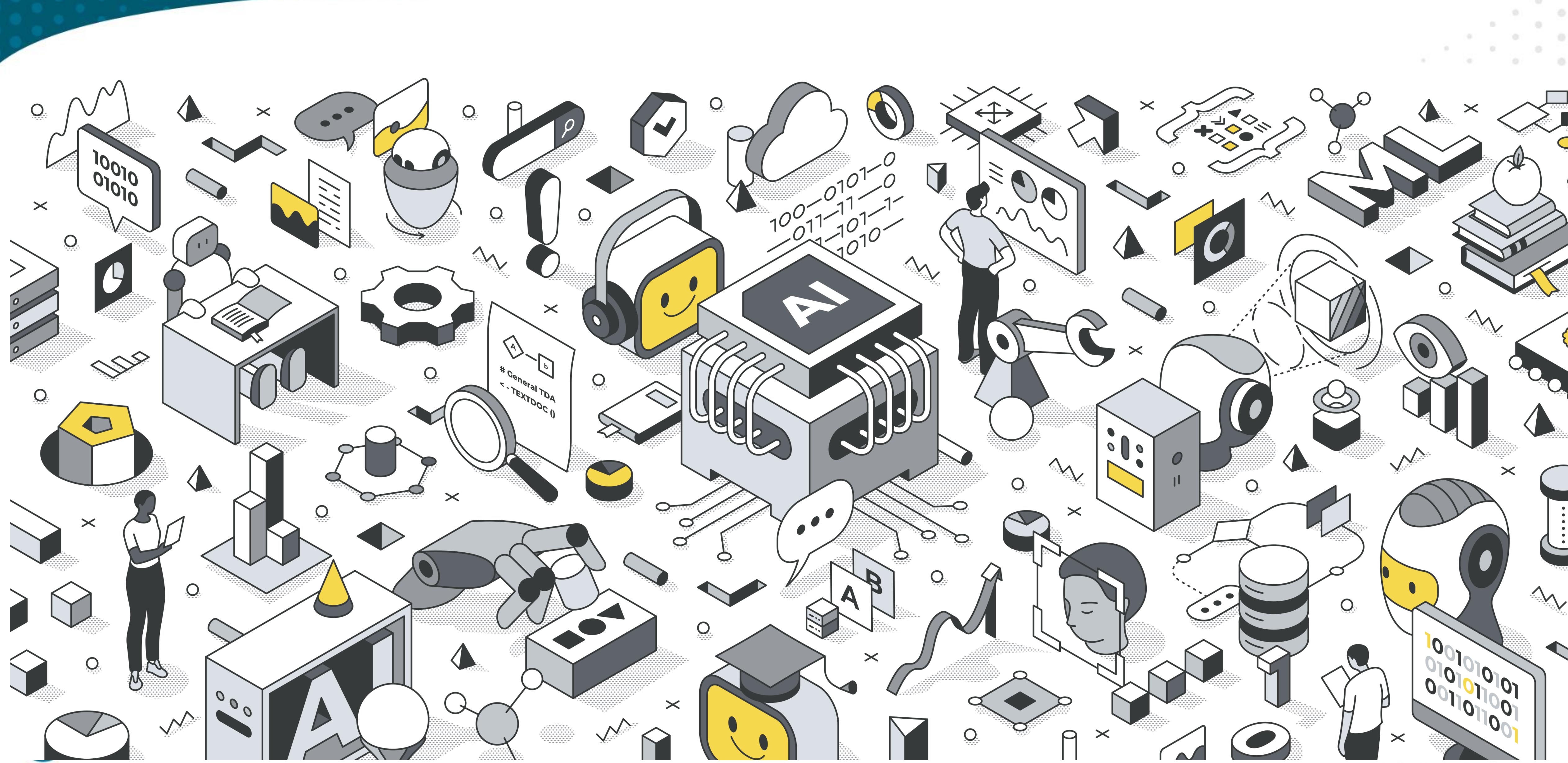
# Why was MCP Created?

- There were no standards
- It's like the Wild Wild West,  
but for AI
- As a developer, where do you  
start in AI when the field looks  
like this...



**Model Context Protocol**

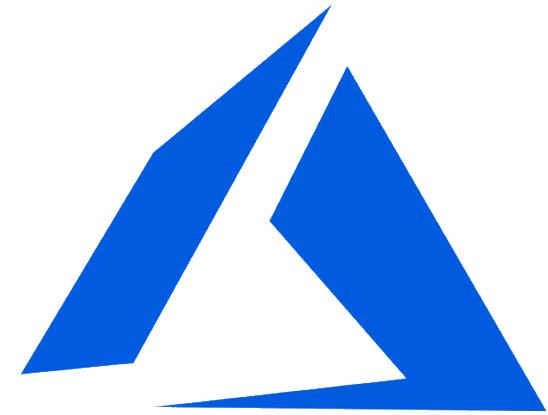
# Why was MCP Created?



# Every Major Player Has Walled Gardens



OpenAI



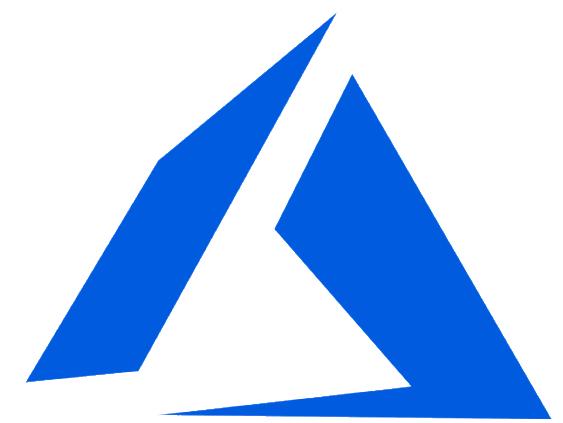
Microsoft  
Azure

Gemini

cohere

Claude

# Who is Officially Supporting the MCP Standard?



Microsoft  
Azure

Gemini

The Claude logo icon is a stylized orange asterisk or star shape with eight points.

Claude

# 50k Github stars in < 6 months

- Break-neck, speed of community adoption

# 50k Github stars in < 6 months

- Break-neck, speed of community adoption
- Only the **top 0.0001 %** of GitHub projects have 50k stars

# 50k Github stars in < 6 months

- Break-neck, speed of community adoption
- Only the **top 0.0001 %** of GitHub projects have 50k stars
- In < 6 months



# WHY?

# I've Seen This Movie Before...

- For some of us, we've seen this movie before

# I've Seen This Movie Before...

- For some of us, we've seen this movie before
- For example, do you remember 3D animated movie from 1998 whose **main character is an Ant?**



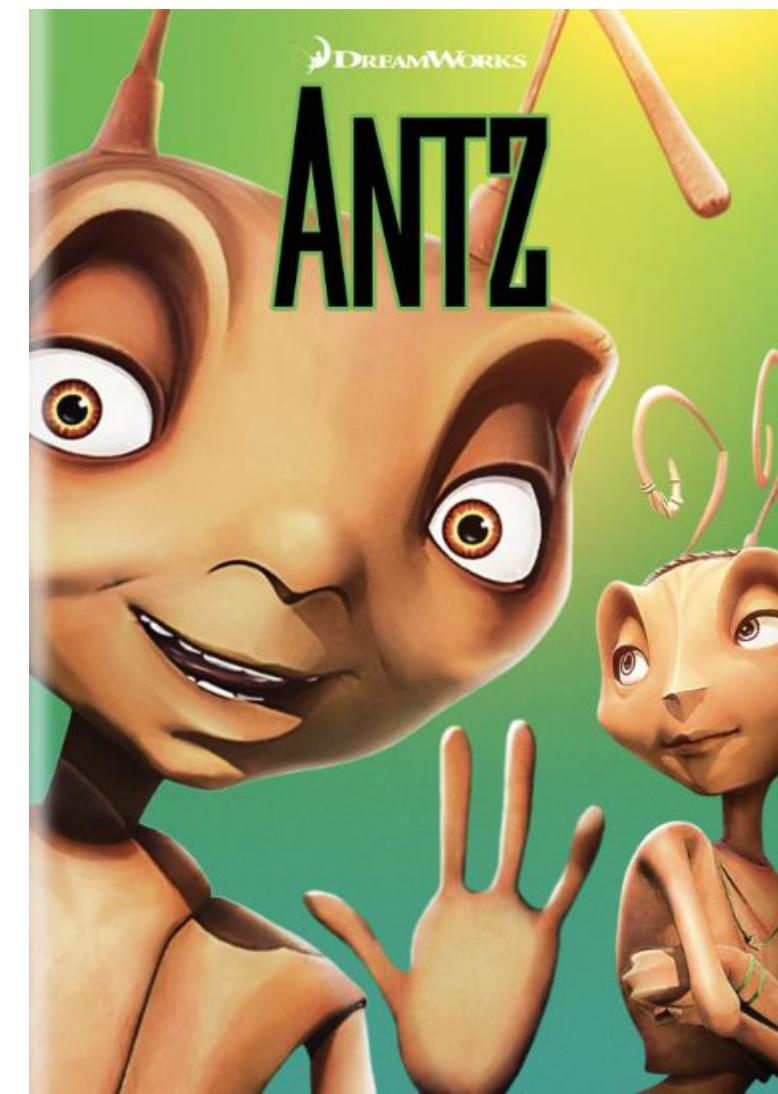
# I've Seen This Movie Before...

- For some of us, we've seen this movie before
- For example, do you remember 3D animated movie from 1998 whose **main character is an Ant?**
- The ant fell in love with the princess ant



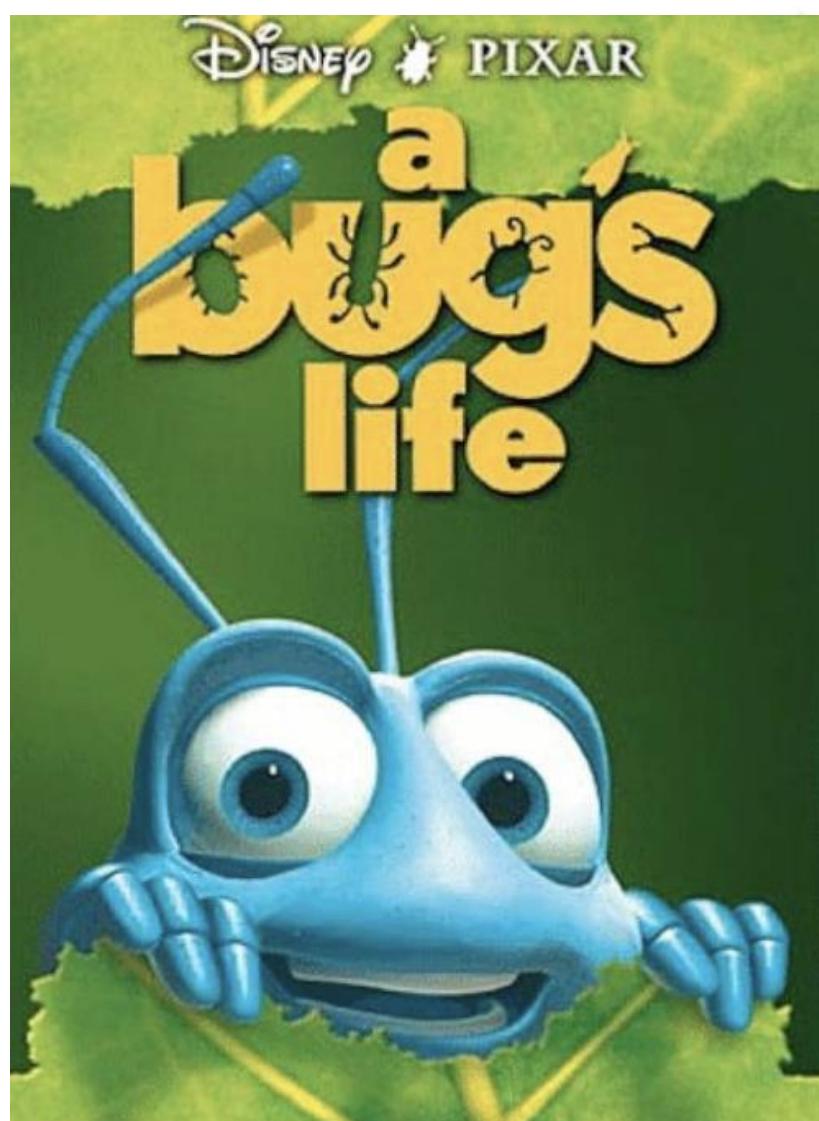
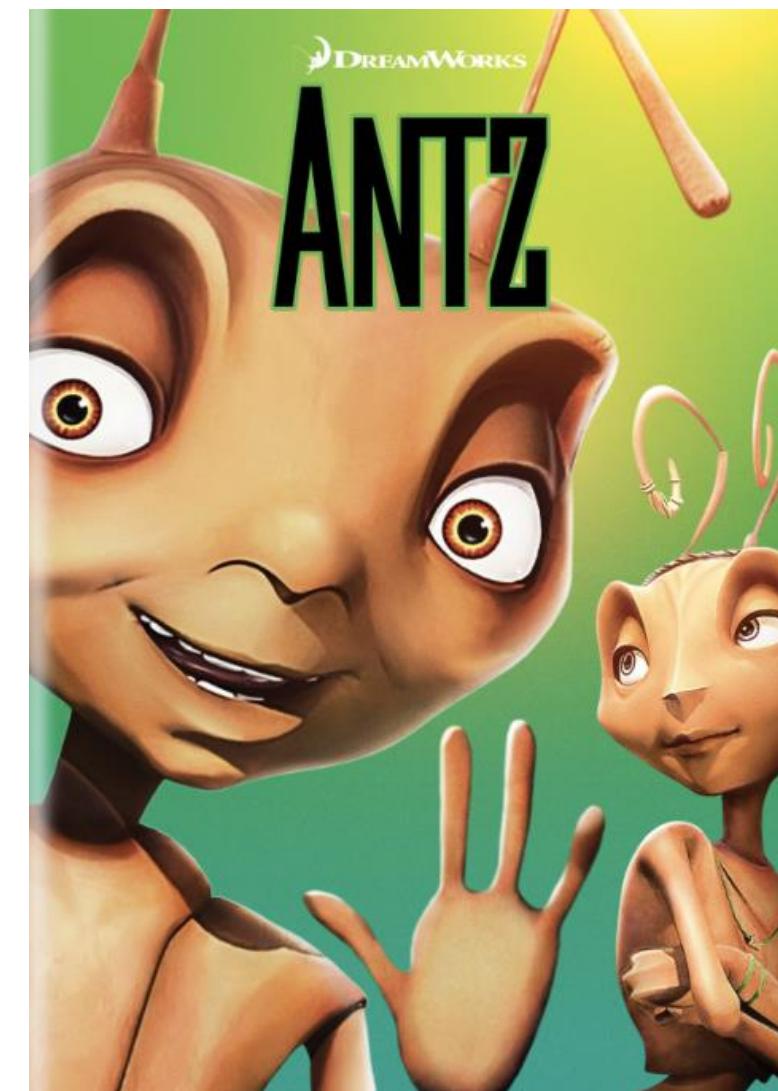
# I've Seen This Movie Before...

- For some of us, we've seen this movie before
- For example, do you remember 3D animated movie from 1998 whose **main character is an Ant?**
- The ant fell in love with the princess ant



# I've Seen This Movie Before...

- For some of us, we've seen this movie before
- For example, do you remember 3D animated movie from 1998 whose **main character is an Ant?**
- The ant fell in love with the princess ant



# Also, Do You Remember...

- For some of us, we've seen this movie before

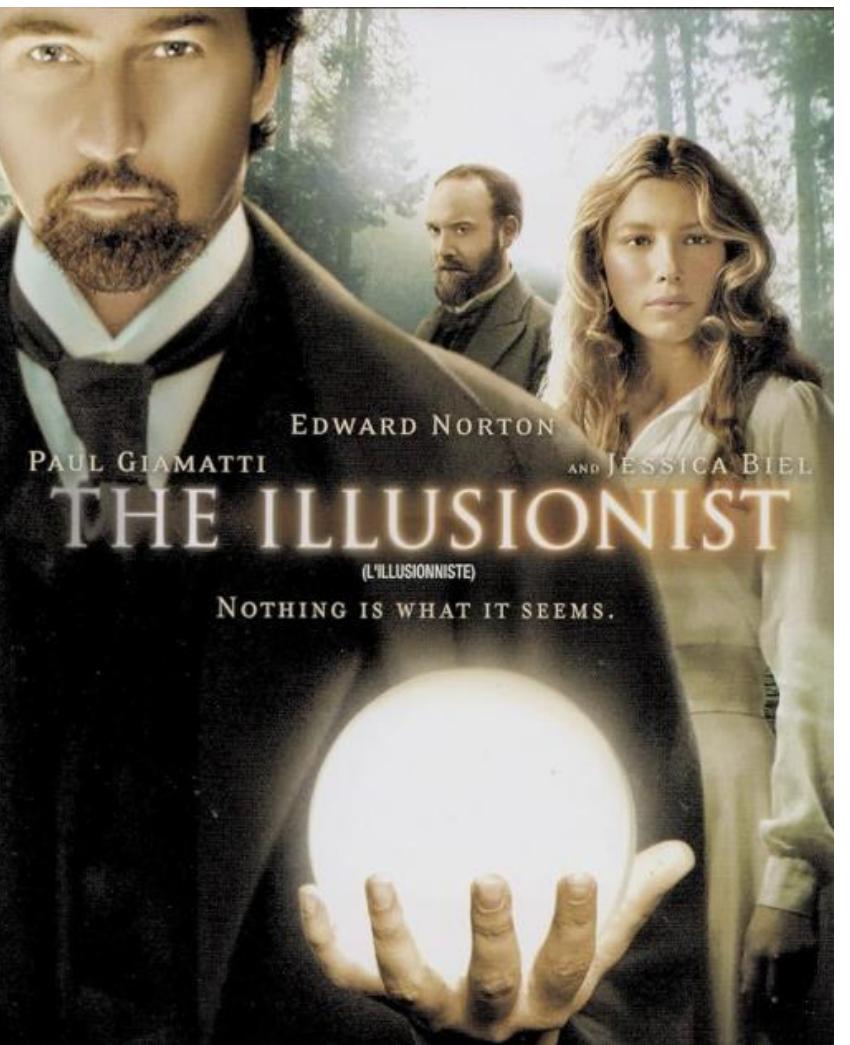
# Also, Do You Remember...

- For some of us, we've seen this movie before
- Do you remember movie from 2006 about **two magicians who were stage performers?**

# Also, Do You Remember...

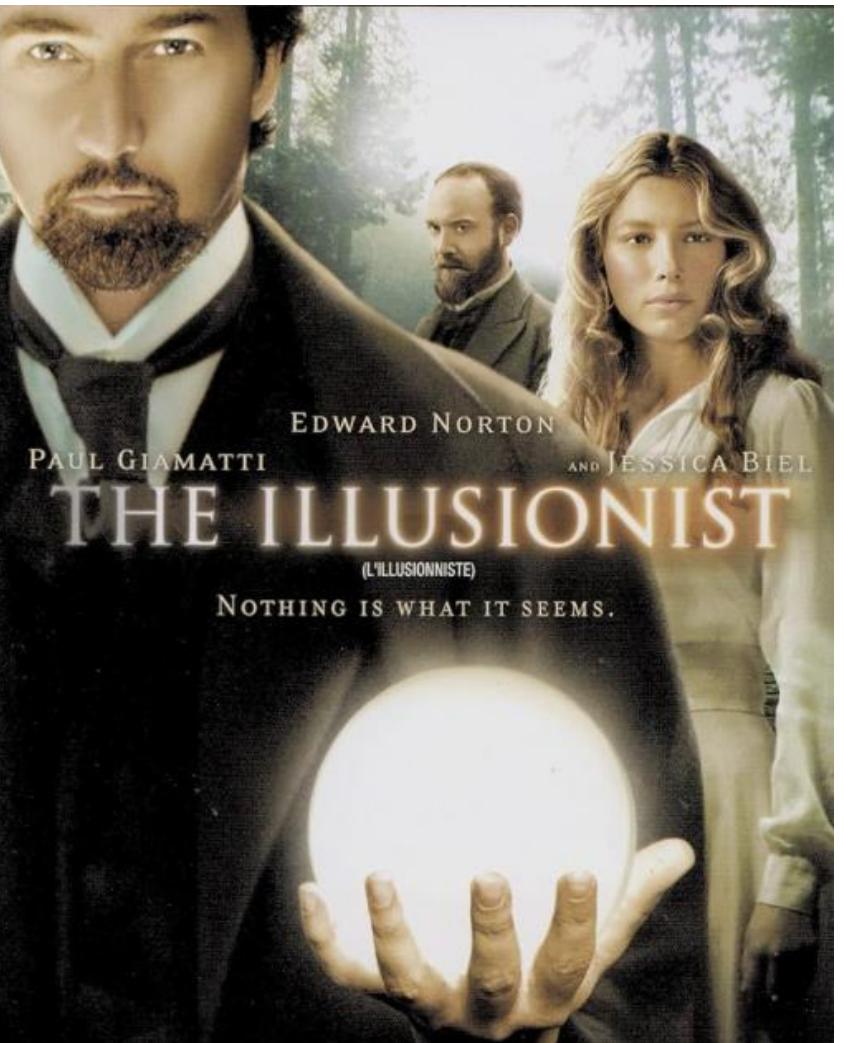
- For some of us, we've seen this movie before
- Do you remember movie from 2006 about **two magicians who were stage performers?**
- The time period was 1800-1900 Victorian era

# Also, Do You Remember...



- For some of us, we've seen this movie before
- Do you remember movie from 2006 about **two magicians who were stage performers?**
- The time period was 1800-1900 Victorian era

# Also, Do You Remember...



- For some of us, we've seen this movie before
- Do you remember movie from 2006 about **two magicians who were stage performers?**
- The time period was 1800-1900 Victorian era



Pearson

# What's the Point?

- The point here is that for some of us, **we've seen this movie before**



# What's the Point?

- The point here is that for some of us, **we've seen this movie before**
- We remember an era of **walled gardens** before the emergence of an industry standard



# What's the Point?

- The point here is that for some of us, **we've seen this movie before**
- We remember an era of **walled gardens** before the emergence of an industry standard
- It was the **WWW**

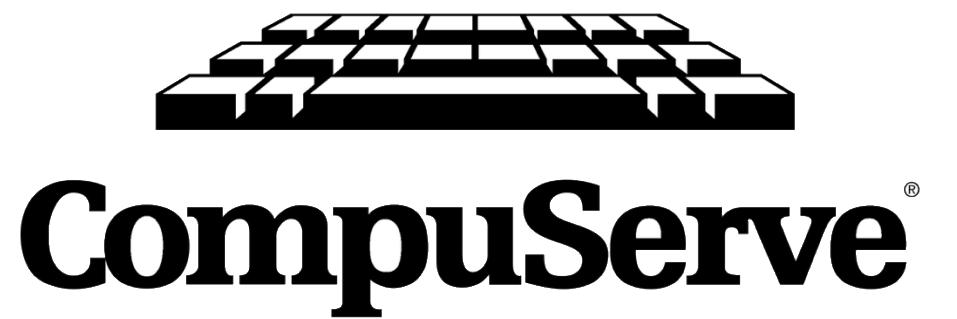


# What's the Point?

- The point here is that for some of us, **we've seen this movie before**
- We remember an era of **walled gardens** before the emergence of an industry standard
- It was the **www**



# What's the Point?



PRODIGY

- In the early days of the public internet, some ISPs had the brilliant idea of creating **walled gardens** of content

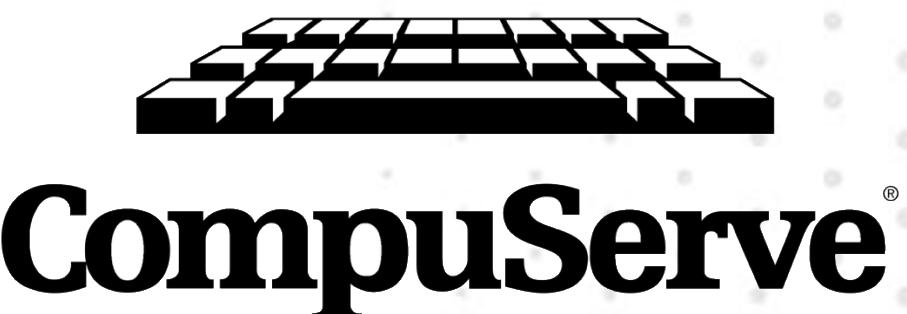
# What's the Point?



- In the early days of the public internet, some ISPs had the brilliant idea of creating **walled gardens** of content
- For example, TV commercials and magazine articles prominently showed their @aol address

# The Walled Garden Strategy

- These services kept everything in their **walled garden**



# The Walled Garden Strategy

- These services kept everything in their **walled garden**
- They all provided internet access



# The Walled Garden Strategy

- These services kept everything in their **walled garden**
- They all provided internet access
- They all created the clients and hosted the servers

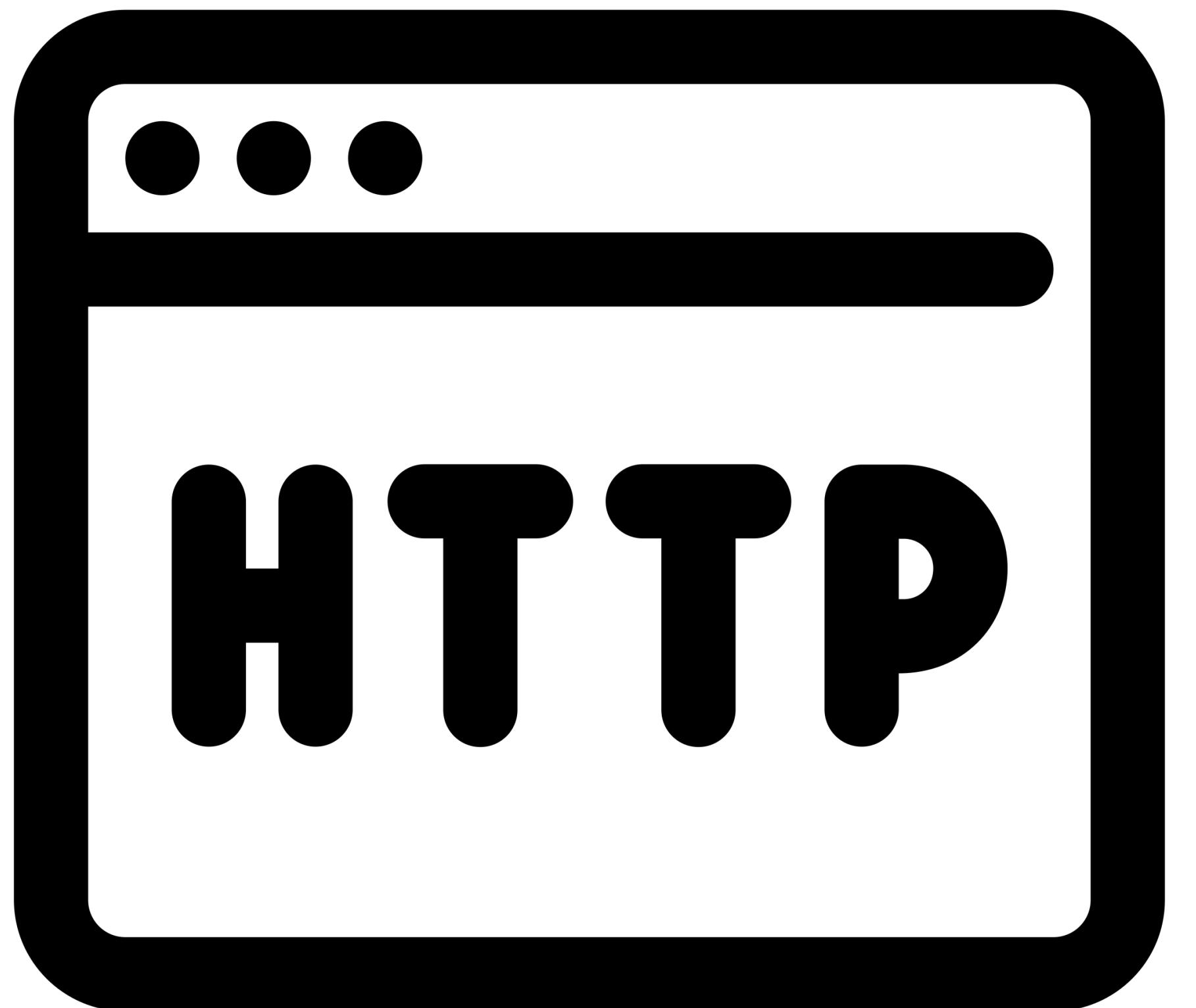


# The Walled Garden Strategy

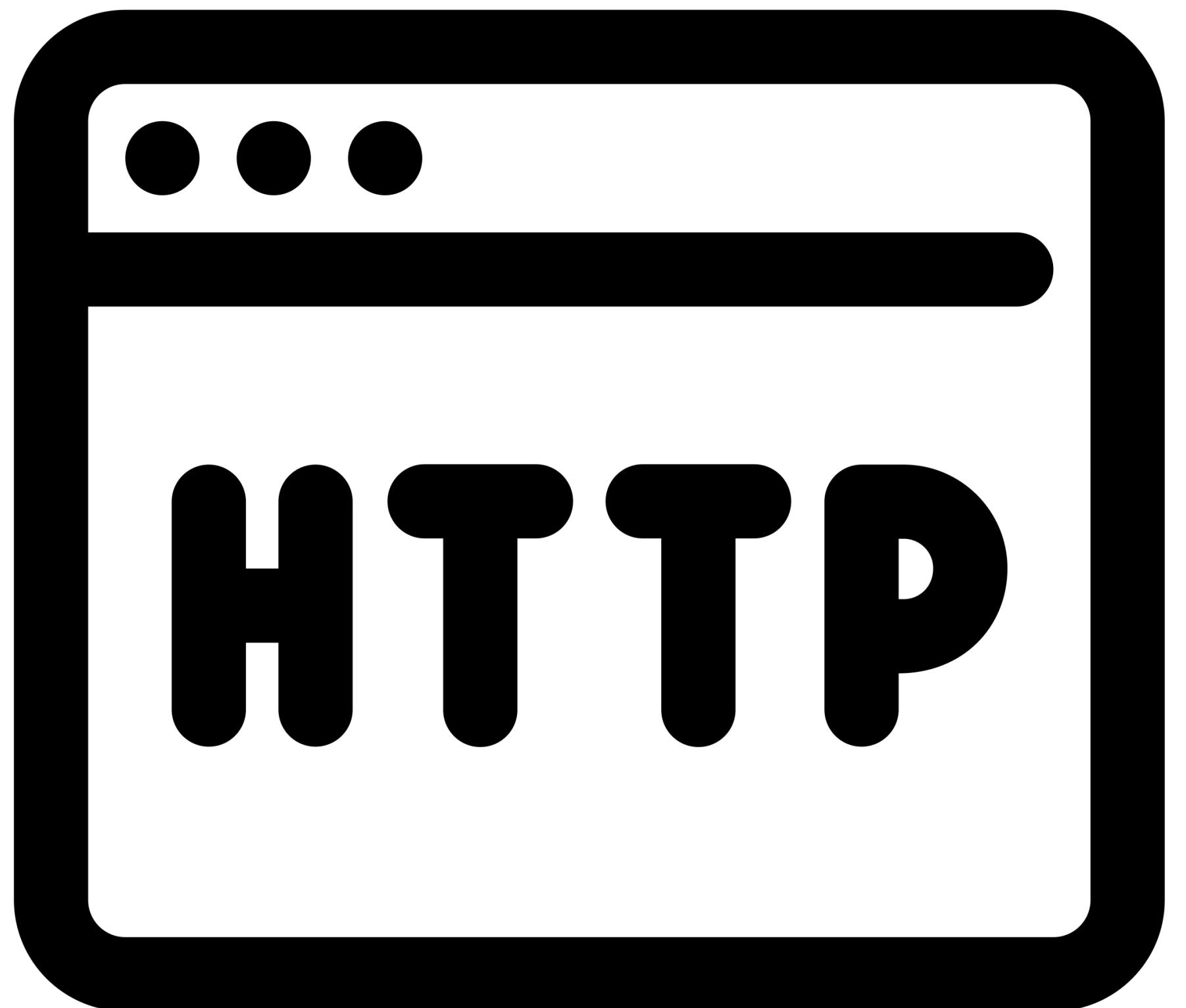
- These services kept everything in their **walled garden**
- They all provided internet access
- They all created the clients and hosted the servers
- They all made their own hyperlink system and addressing



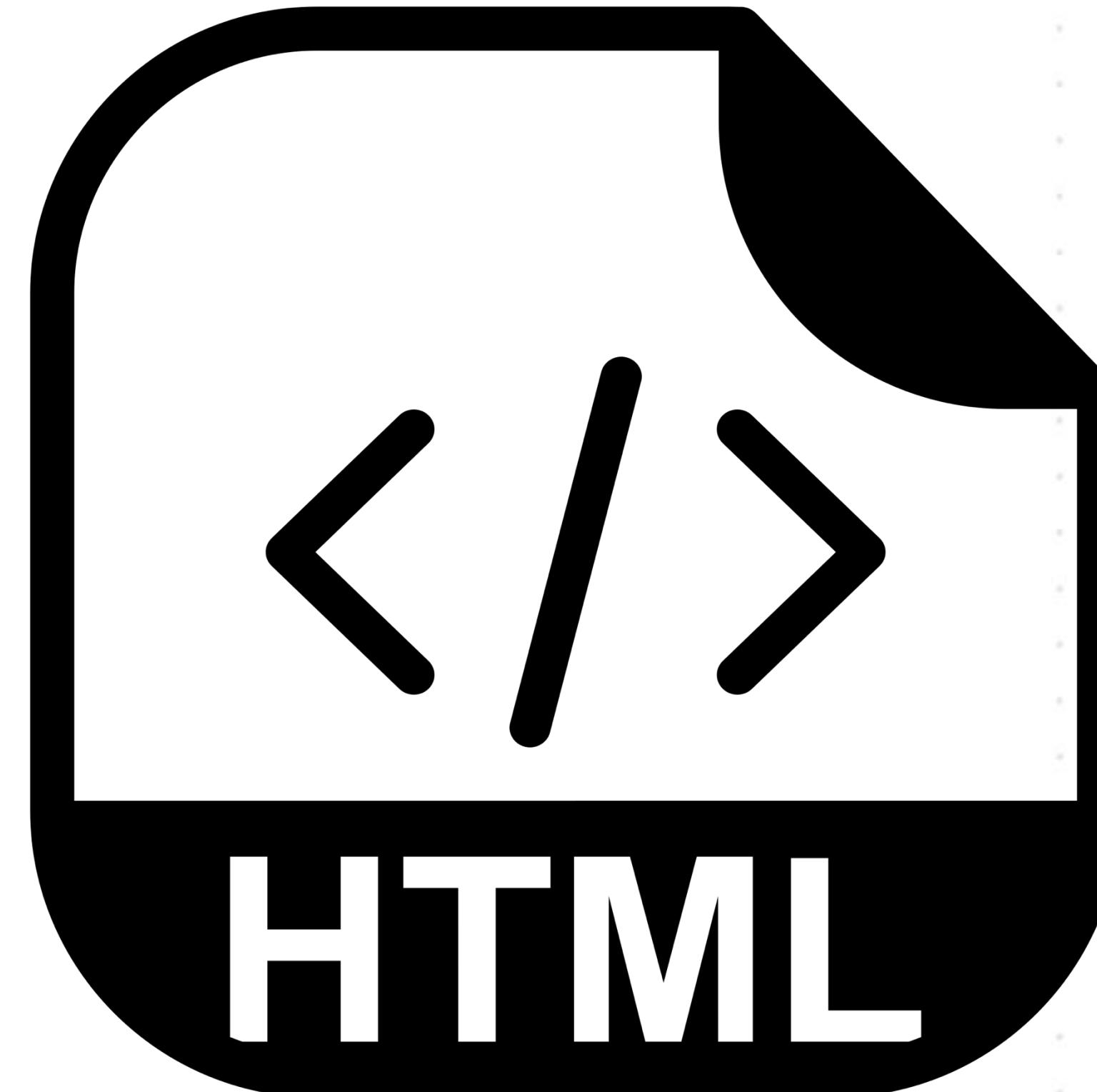
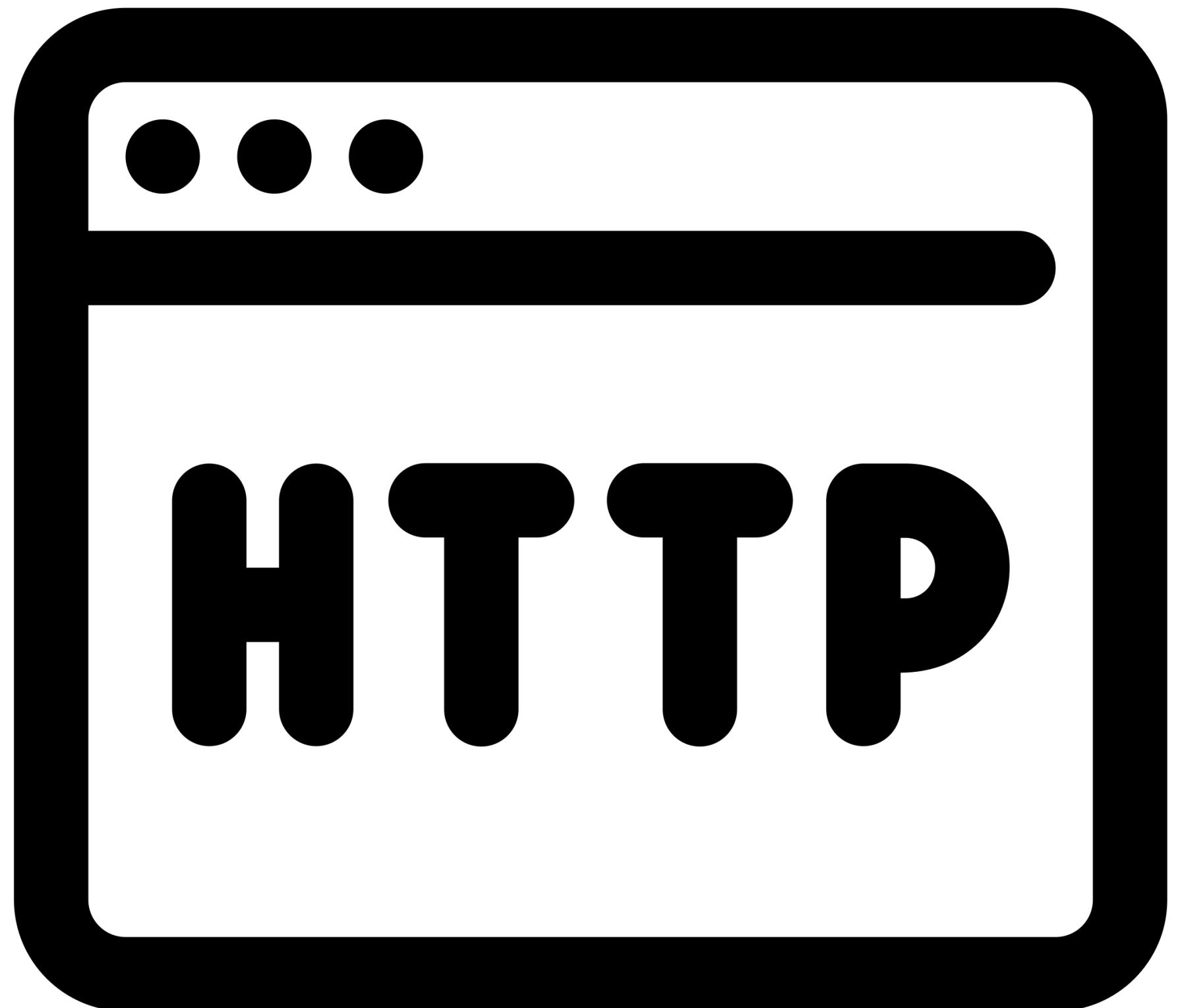
**And they all failed**



# Open Standards



# Open Standards



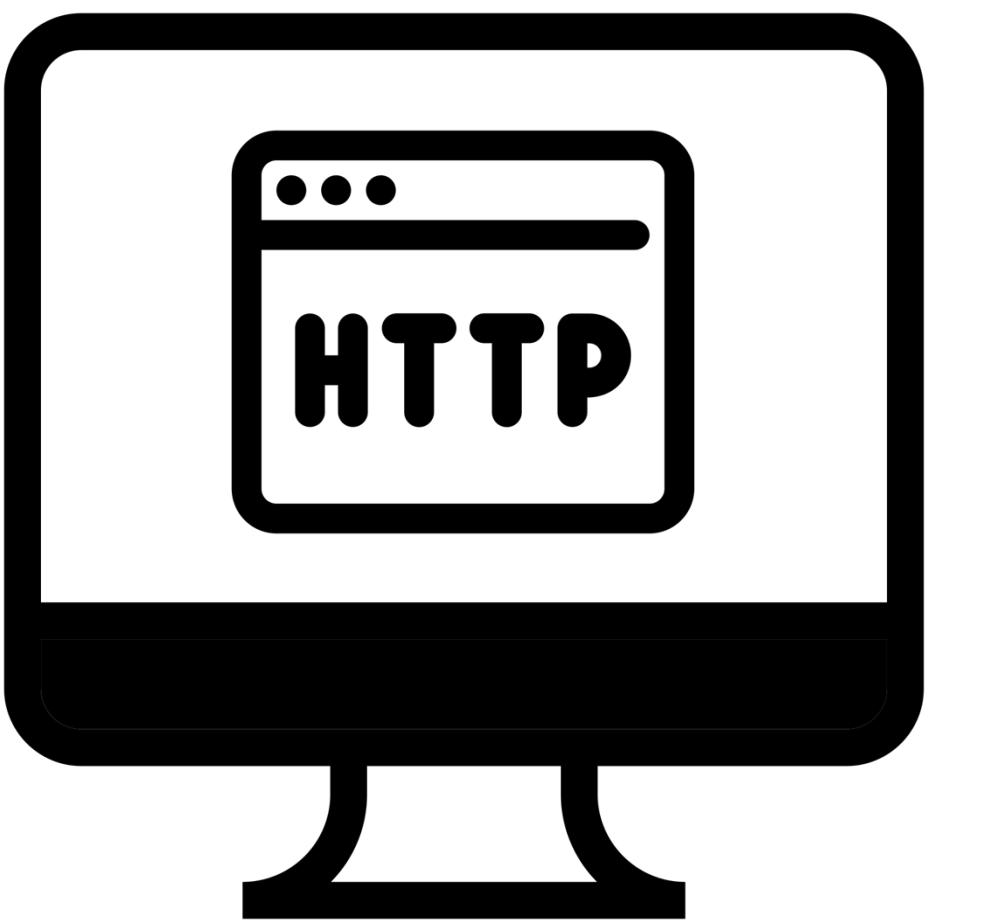
Disruptive technology

# How Did it Work?

- Simple client/server architecture

# How Did it Work?

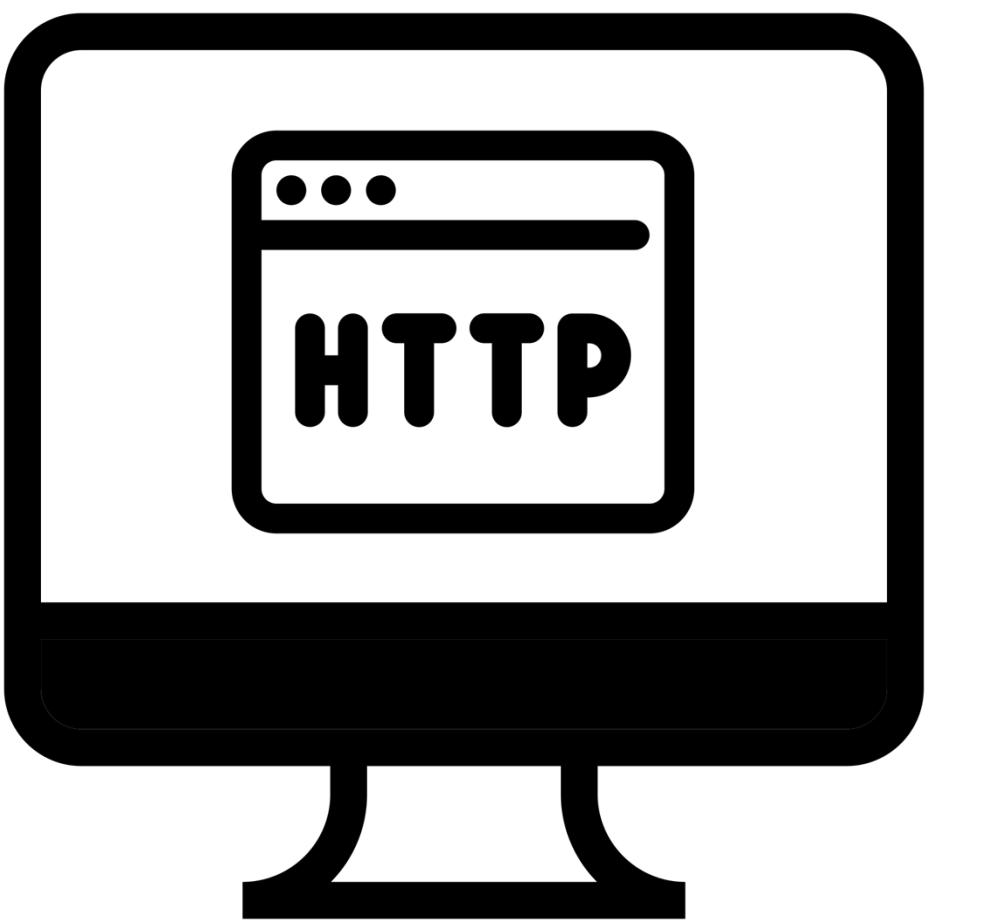
- Simple client/server architecture



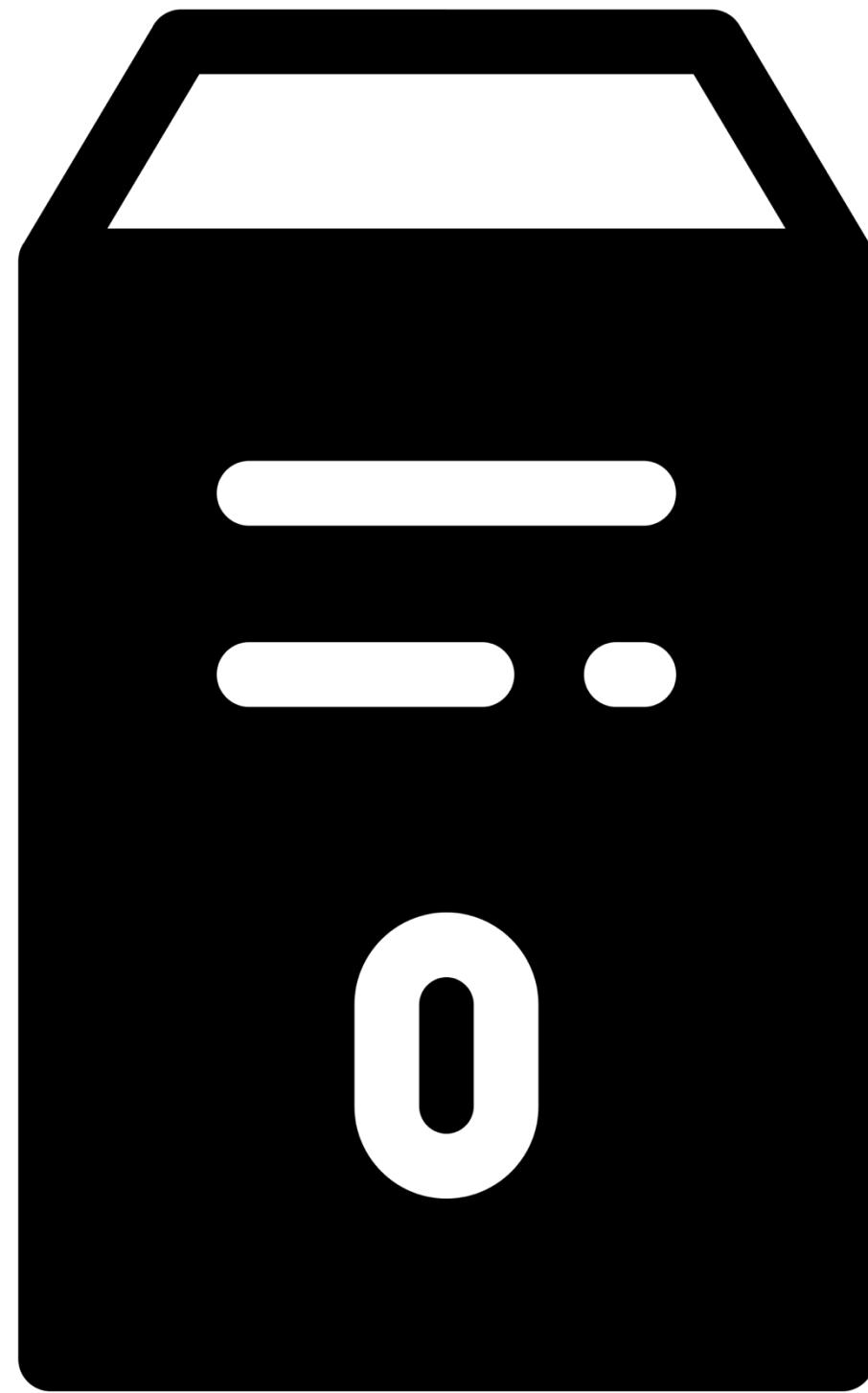
Netscape  
MS Internet  
Explorer

# How Did it Work?

- Simple client/server architecture



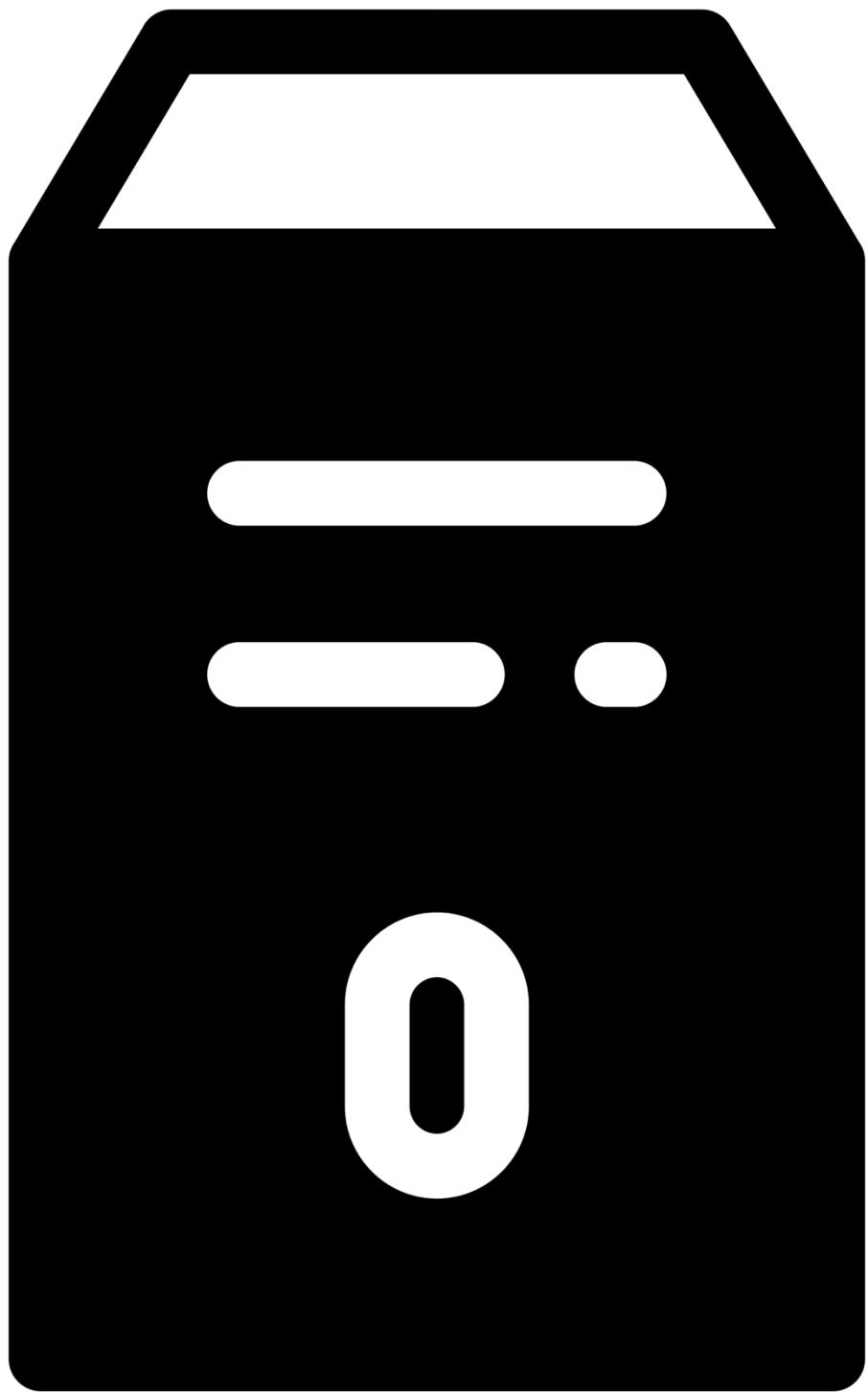
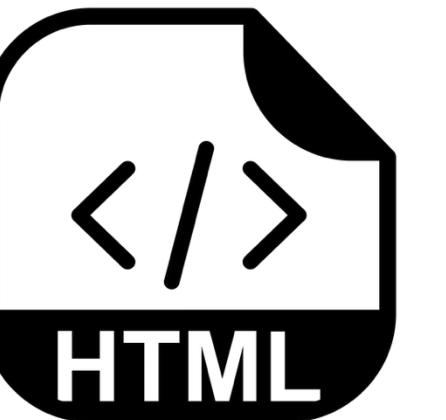
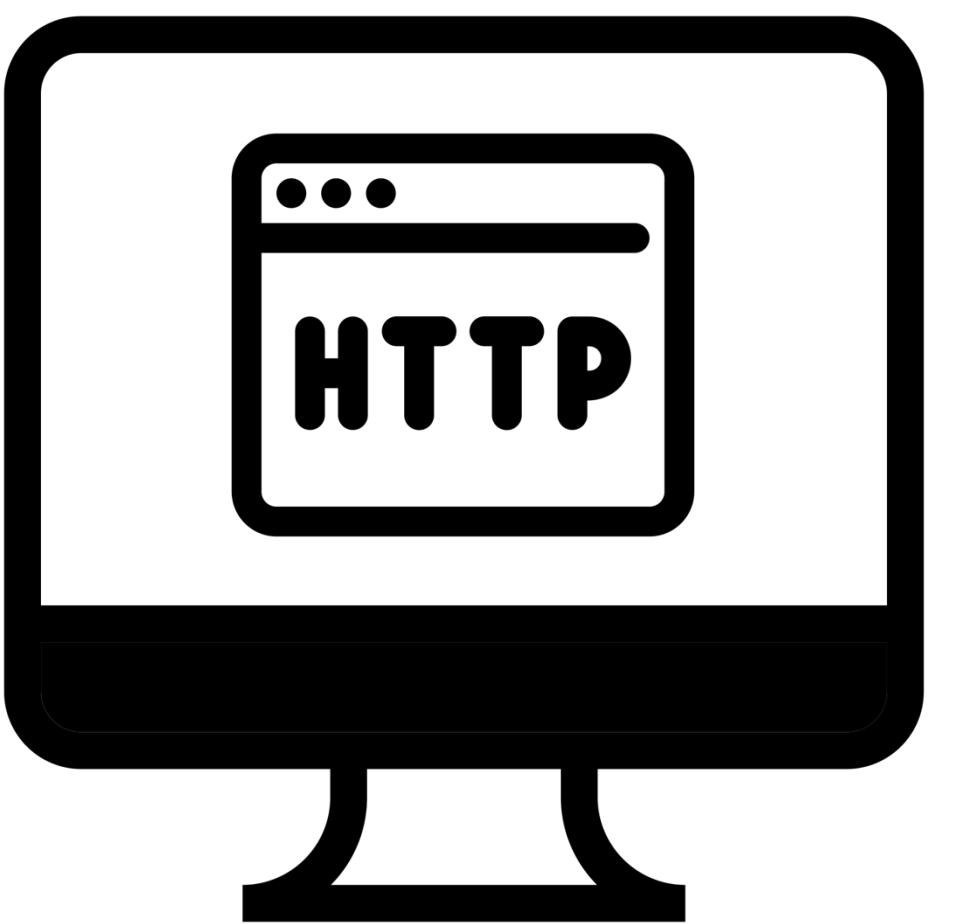
Netscape  
MS Internet  
Explorer



Apache Webserver  
MS IIS

# How Did it Work?

- Simple client/server architecture



Netscape  
MS Internet  
Explorer

Apache Webserver  
MS IIS

# What's the Point of that History Lesson?

**WHY?**

# What's the Point of that History Lesson?

- **WHY** did the MCP Server project on Github get 50k stars in < 6 months?



# What's the Point of that History Lesson?

- **WHY** did the MCP Server project on Github get 50k stars in < 6 months?
- **WHY** did OpenAI, Google, and Microsoft announce support for it?



# What's the Point of that History Lesson?

- **WHY** did the MCP Server project on Github get 50k stars in < 6 months?
- **WHY** did OpenAI, Google, and Microsoft announce support for it?
- **WHY** is the industry as a whole is adopting it?

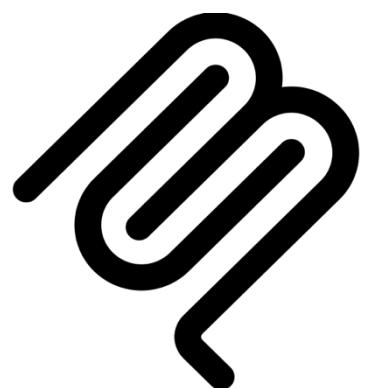
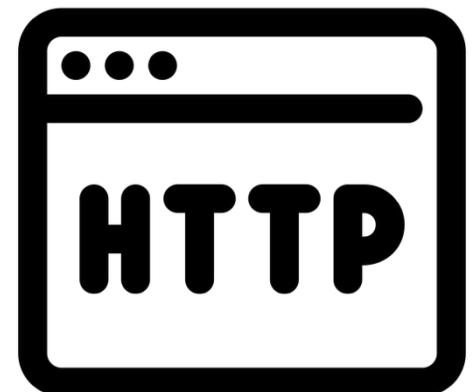


# We've seen this movie before

- If you like “*The Illusionist*”, then you’re going to love, “*The Prestige*”

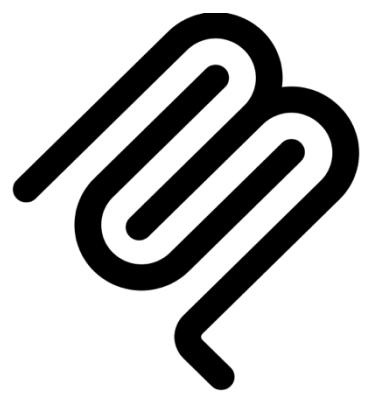
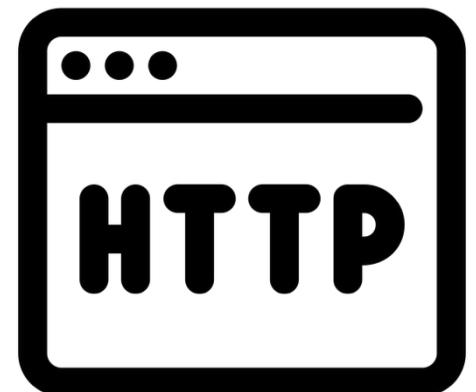
# We've seen this movie before

- If you like “*The Illusionist*”, then you’re going to love, “*The Prestige*”
- If you like HTTP, then you’re going to love MCP



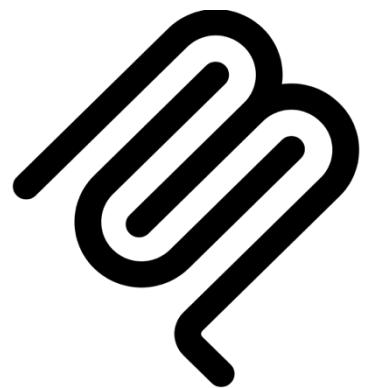
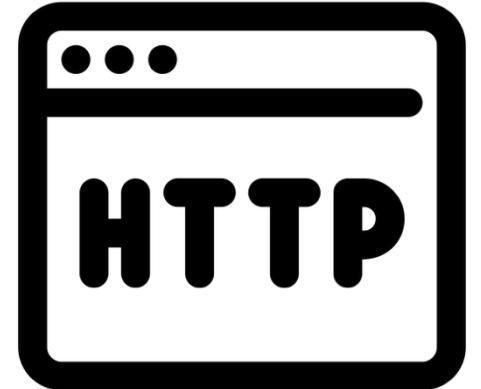
# We've seen this movie before

- If you like “*The Illusionist*”, then you’re going to love, “*The Prestige*”
- If you like HTTP, then you’re going to love MCP
- How Does MCP Work?



# We've seen this movie before

- If you like “*The Illusionist*”, then you’re going to love, “*The Prestige*”
- If you like HTTP, then you’re going to love MCP
- How Does MCP Work?
- Simple client/server architecture



# MCP Client

- The Host/**Client** Connects to MCP Servers



MCP Client

# MCP Client

- The Host/**Client** Connects to MCP Servers
- Usually bundles with an AI model (LLM)



MCP Client

# MCP Client

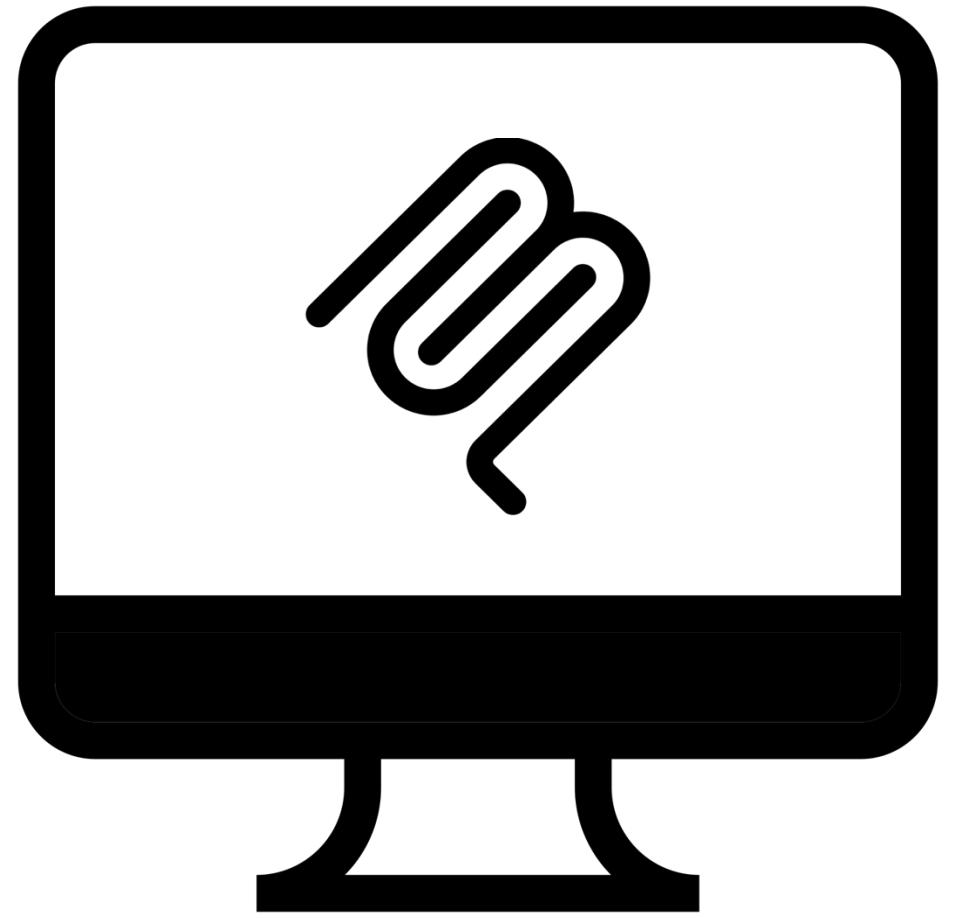
- The Host/**Client** Connects to MCP Servers
- Usually bundles with an AI model (LLM)
- Provides the user interface to the MCP Server



MCP Client

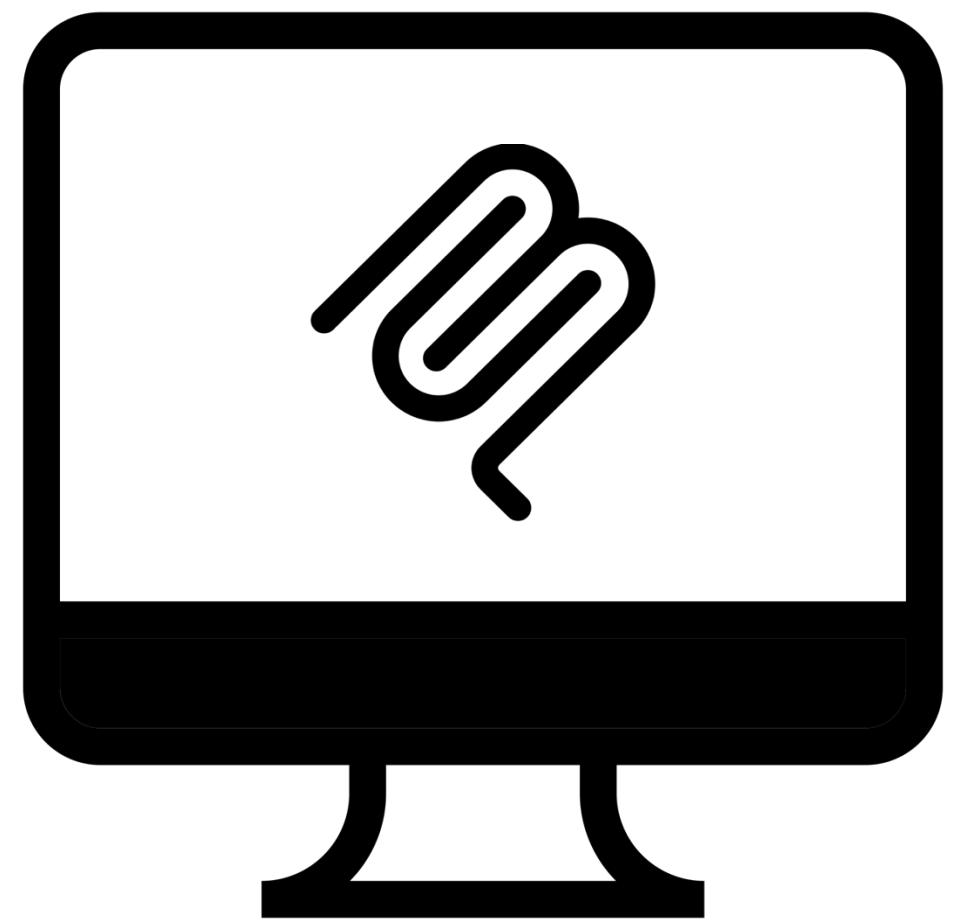
# Example MCP Client Apps

- ChatGPT Playground



MCP Client

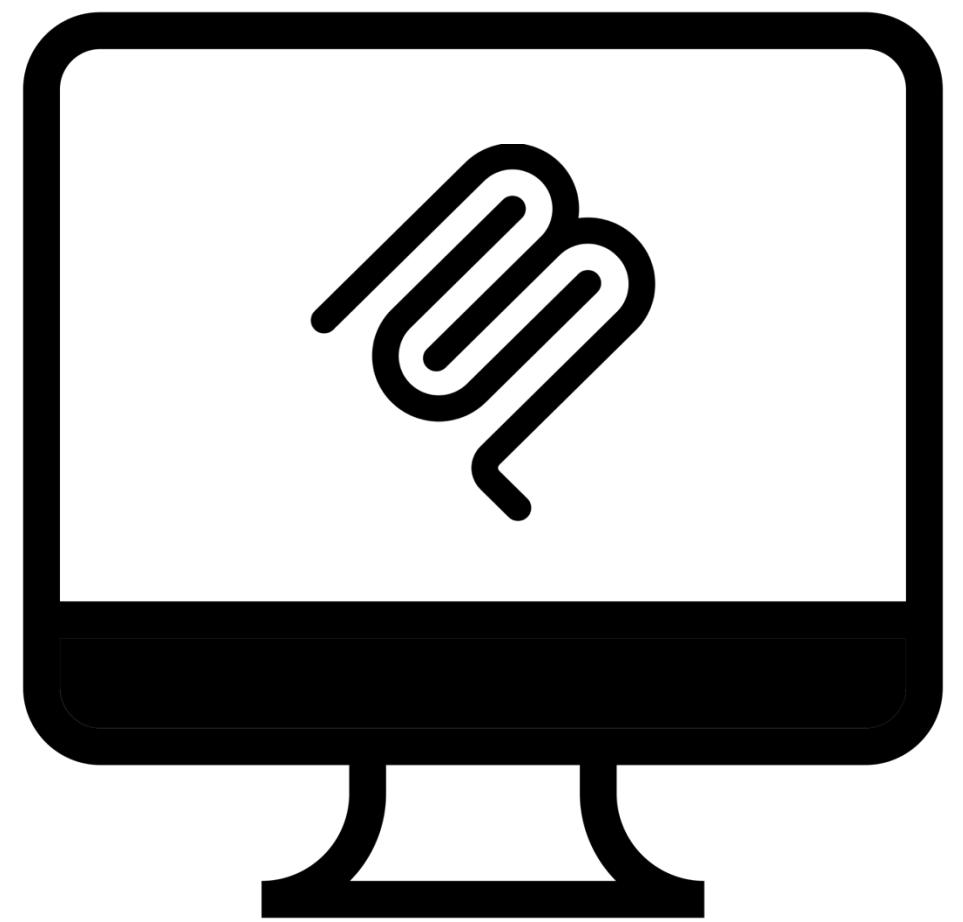
# Example MCP Client Apps



MCP Client

- ChatGPT Playground
- Claude Desktop

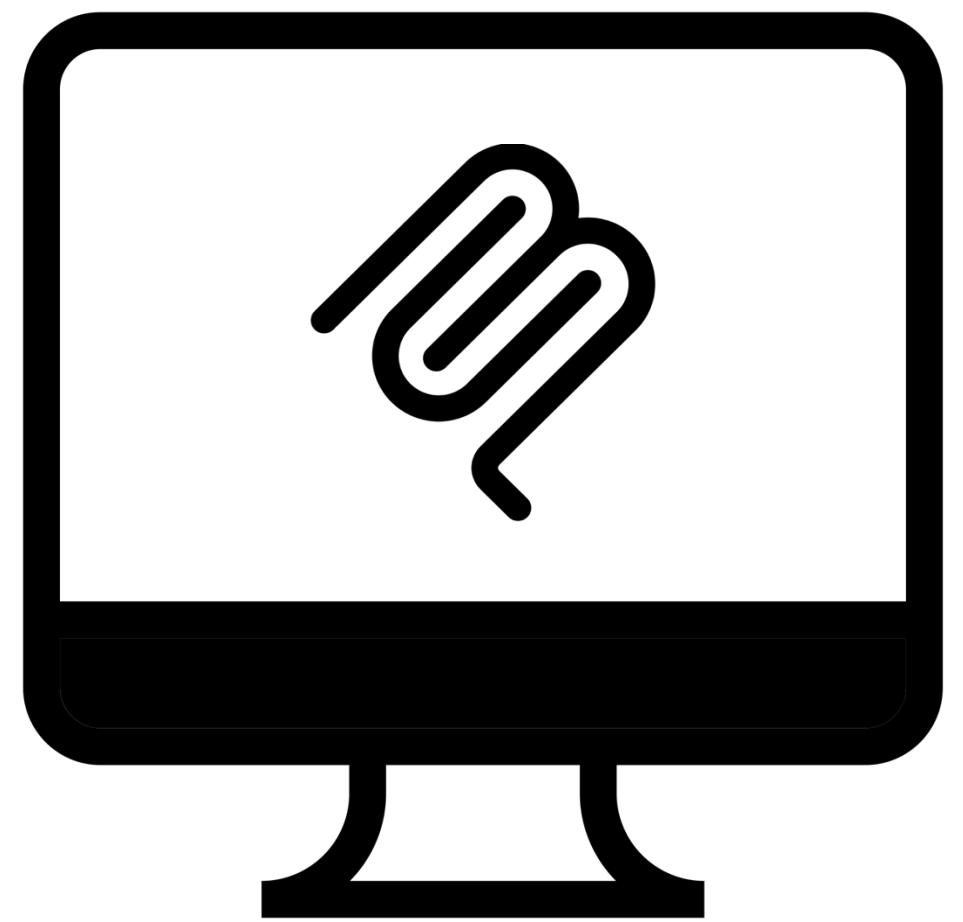
# Example MCP Client Apps



MCP Client

- ChatGPT Playground
- Claude Desktop
- Postman

# Example MCP Client Apps



MCP Client

- ChatGPT Playground
- Claude Desktop
- Postman
- MCP Inspector

# Example MCP Client Apps



# ChatGPT Playground

Prompts Playground - OpenAI API

BT Authors / O'Reilly Course AI Agents

Playground Dashboard Docs API reference

PLAYGROUND Prompts Images Realtime Assistants TTS

Prompts Your prompts Save

Add tools from remote MCP servers

Select from a list of popular server or connect to a new one

+ Add new Zapier Shopify

Intercom Stripe Plaid

Square Cloudflare Browser HubSpot

Pipedream PayPal DeepWiki (Devin)

onversation will appear here

3.5s ↑ 29t ↓ 79t

Cookbook Forum Help

Add messages to prompt +

Chat with your prompt... Auto-clear ↑

The screenshot shows the ChatGPT Playground interface. The main navigation bar includes 'Playground', 'Dashboard', 'Docs', and 'API reference'. On the left, there's a sidebar with links for 'PLAYGROUND', 'Prompts' (which is selected), 'Images', 'Realtime', 'Assistants', and 'TTS'. The main content area has tabs for 'Your prompts' and 'Save'. A prominent modal window titled 'Add tools from remote MCP servers' lists various services with their logos: Zapier, Shopify, Intercom, Stripe, Plaid, Square, Cloudflare Browser, HubSpot, Pipedream, PayPal, and DeepWiki (Devin). Below the modal, there's a message placeholder 'onversation will appear here' and some performance metrics: '3.5s ↑ 29t ↓ 79t'. At the bottom, there are buttons for 'Cookbook', 'Forum', and 'Help', along with fields for 'Add messages to prompt' and a 'Chat with your prompt...' input field with an 'Auto-clear' button.

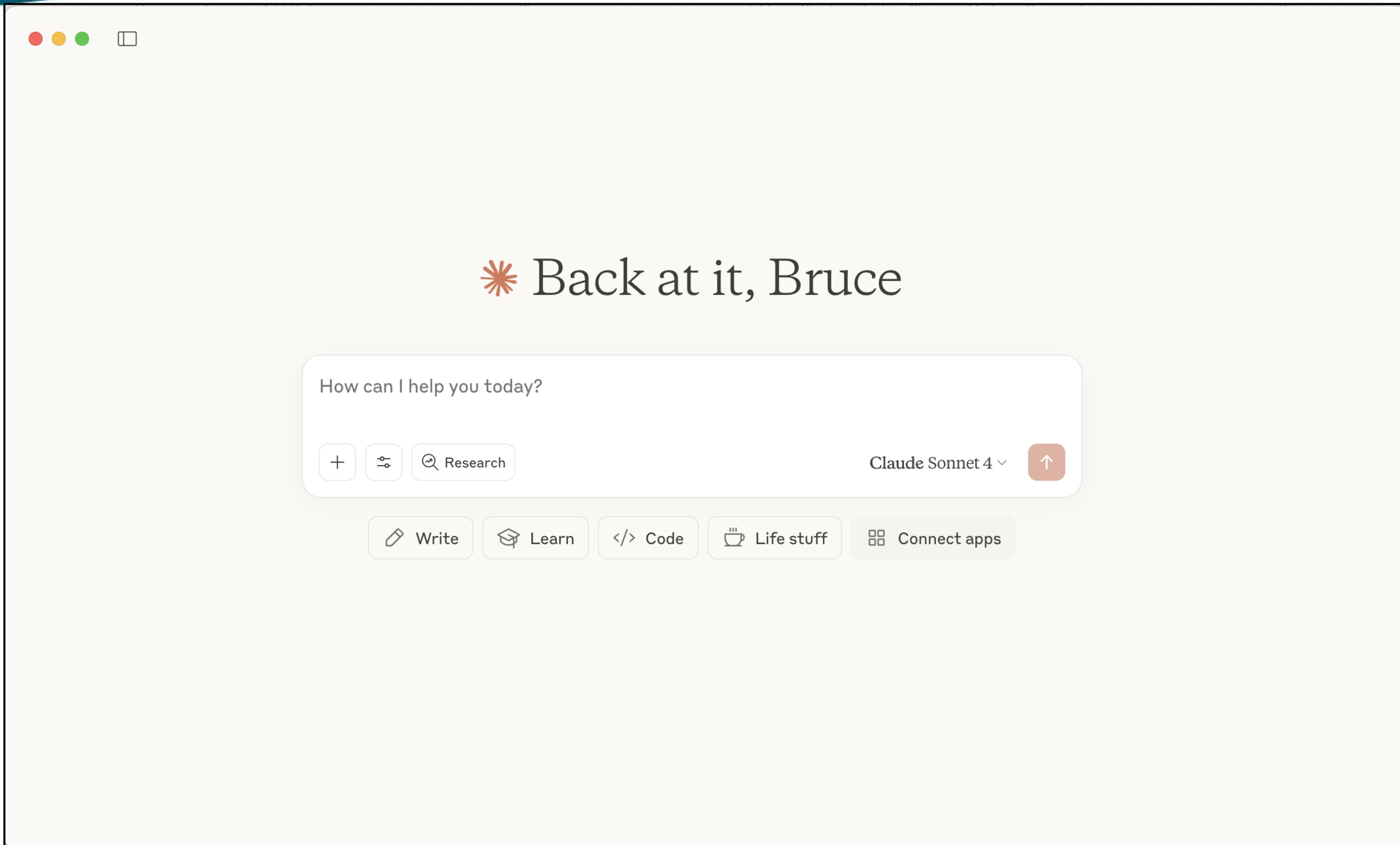
# ChatGPT Playground

The screenshot shows the OpenAI API playground interface. The top navigation bar includes 'Playground', 'Dashboard', 'Docs', 'API reference', and a user profile icon. The left sidebar has sections for 'PLAYGROUND' (Prompts, Images, Realtime, Assistants, TTS), 'Cookbook', 'Forum', and 'Help'. The main area is titled 'Prompts' with tabs for 'Your prompts' and 'Save'. A central modal window is open, titled 'Add tools from remote MCP servers', with the sub-instruction 'Select from a list of popular server or connect to a new one'. It lists nine tools in a 3x3 grid: Add new (plus icon), Zapier (orange square with 'zapier'), Shopify (green shopping bag), Intercom (chat icon), Stripe (blue 'S'), Plaid (grid icon), Square (square icon), Cloudflare Browser (cloud icon), HubSpot (orange hexagon), Pipedream (green 'p'), PayPal (blue 'P'), and DeepWiki (Devin) (black hexagon). Below the modal is a text input field 'Chat with your prompt...' and a message history area.

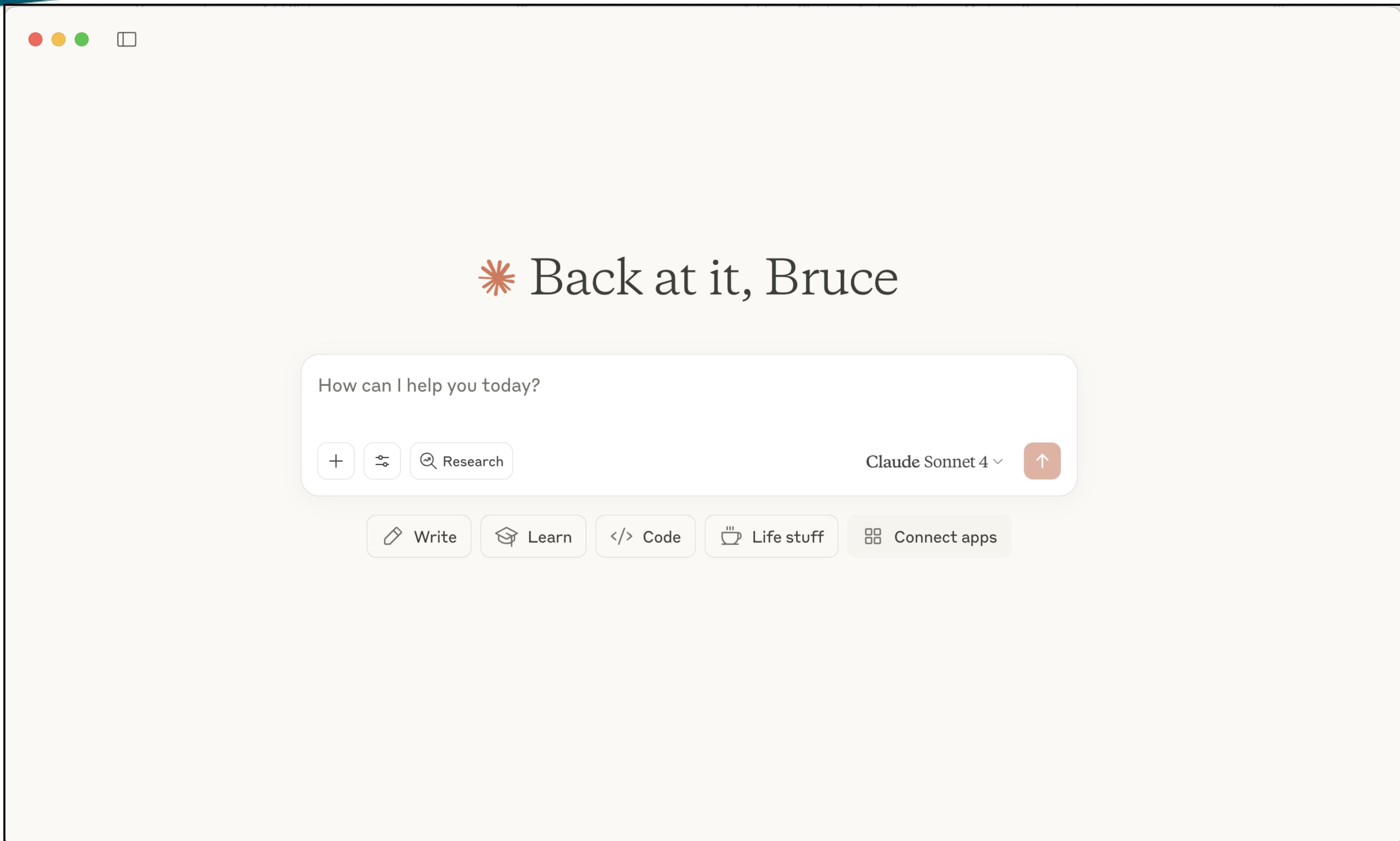


Remote

# Claude Desktop



# Claude Desktop

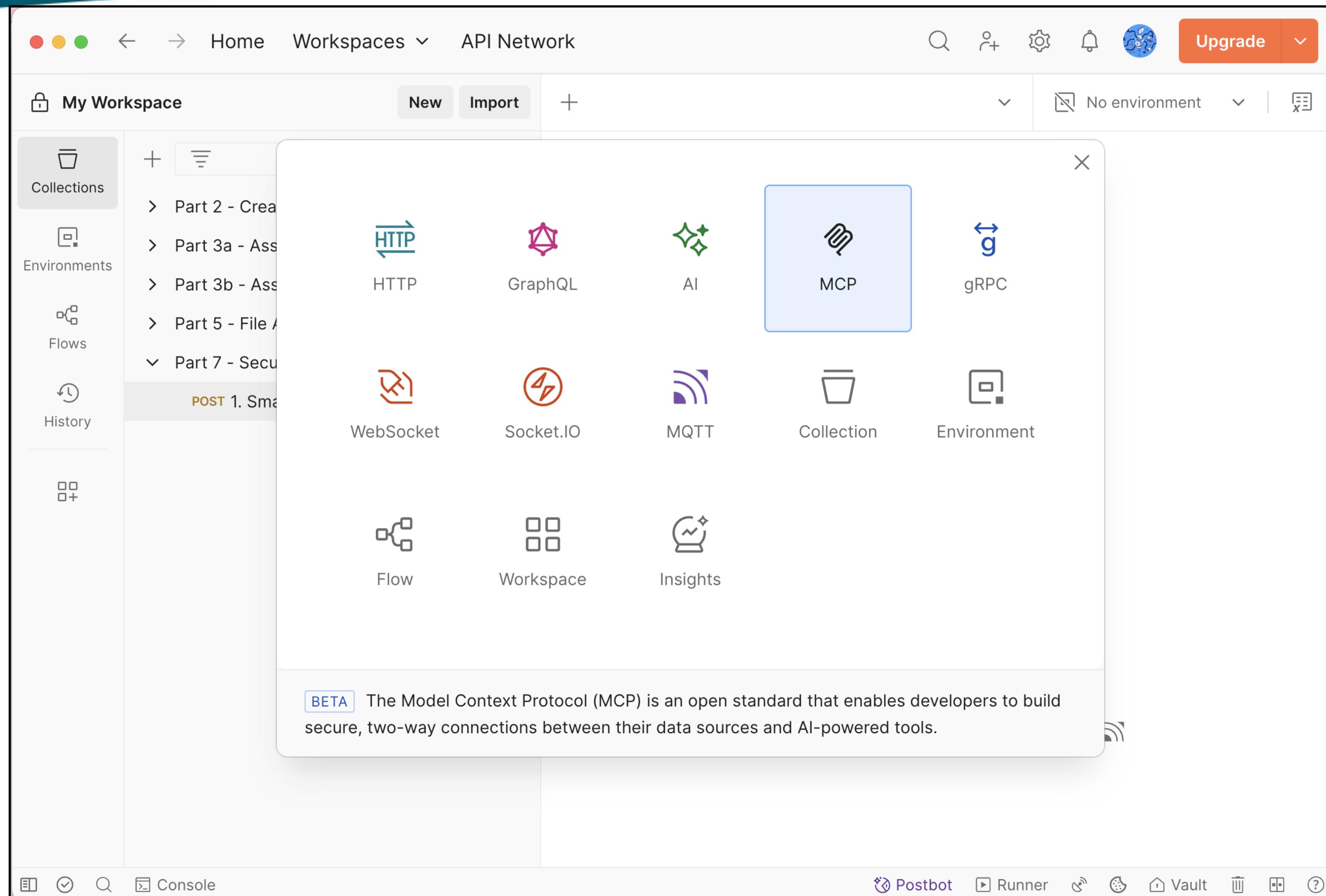


Local

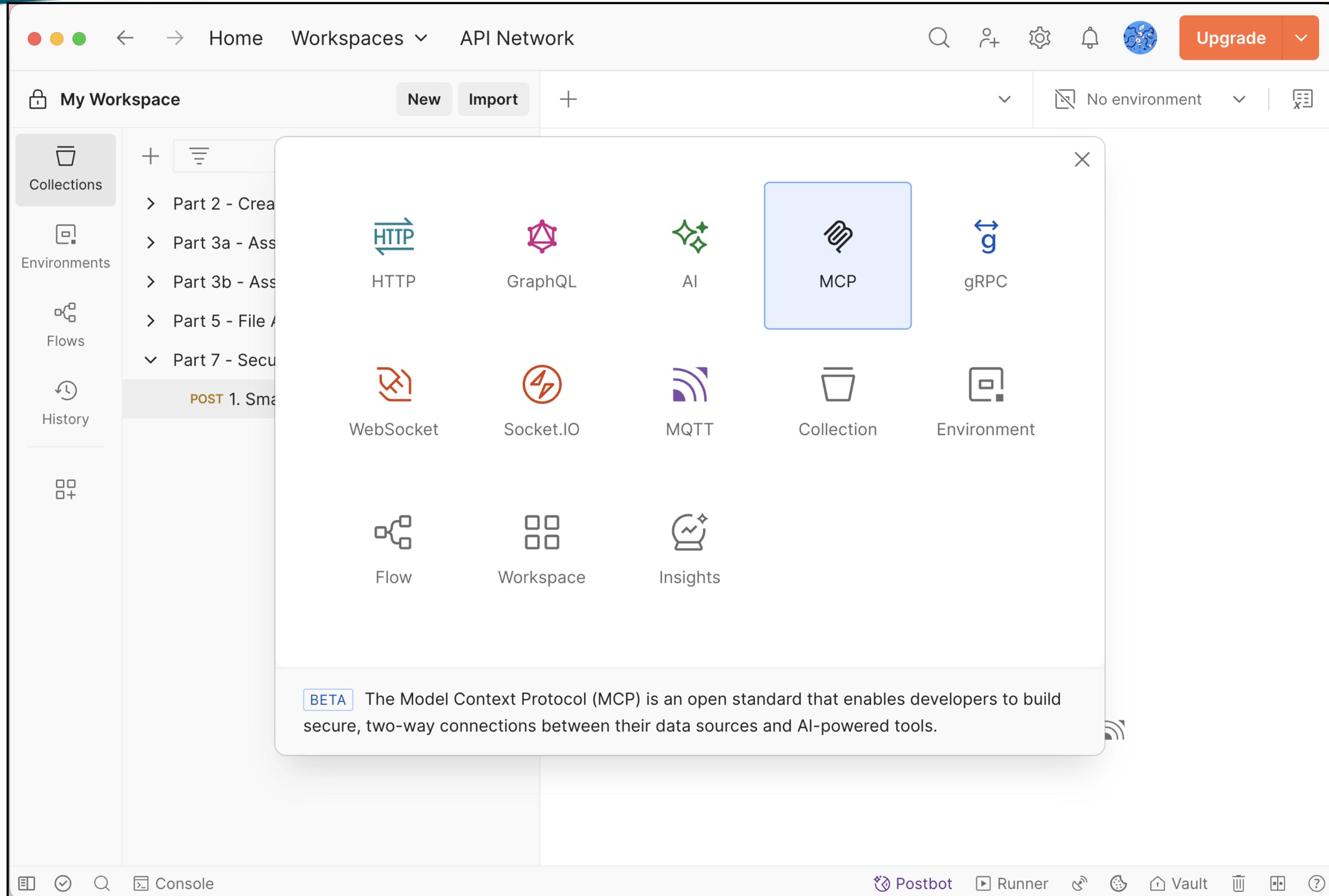


Remote

# Postman



# Postman



The screenshot shows the Postman application interface. On the left, there's a sidebar with 'My Workspace' containing 'Collections', 'Environments', 'Flows', and 'History'. The main area displays various API testing tools: 'HTTP', 'GraphQL', 'AI', 'MCP' (which is highlighted with a blue box), 'gRPC', 'WebSocket', 'Socket.IO', 'MQTT', 'Collection', 'Environment', 'Flow', 'Workspace', and 'Insights'. A tooltip for the MCP icon states: 'BETA The Model Context Protocol (MCP) is an open standard that enables developers to build secure, two-way connections between their data sources and AI-powered tools.' At the bottom, there are tabs for 'Console', 'Postbot', 'Runner', 'Vault', and help icons.



Local



Remote

# MCP Inspector

**MCP Inspector v0.14.2**

Transport Type

STDIO

Command

Command

Arguments

Arguments (space-separated)

> Environment Variables

Server Entry Servers File

> Configuration

▷ Connect

Disconnected

System ▾

?

!

!

Connect to an MCP server to start inspecting

Need to configure authentication? [Open Auth Settings](#)

**History**

No history yet

**Server Notifications**

No notifications yet

# MCP Inspector

MCP Inspector v0.14.2

Transport Type

STDIO

Command

Command

Arguments

Arguments (space-separated)

> Environment Variables

Server Entry Servers File

> Configuration

Connect

Disconnected

History

No history yet

System

?

?

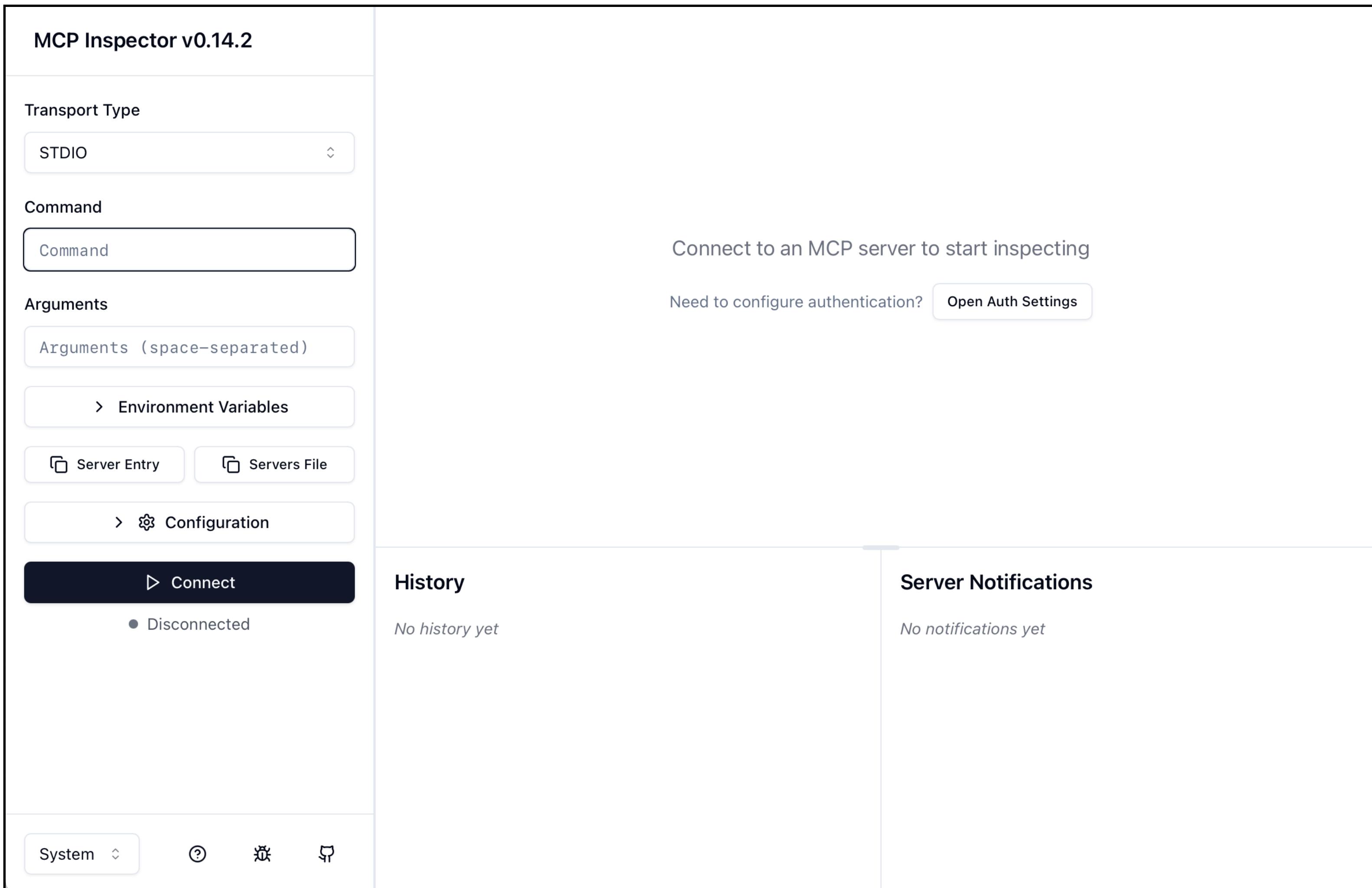
?

Need to configure authentication? Open Auth Settings

Connect to an MCP server to start inspecting

Server Notifications

No notifications yet



Local



Remote

# In This Course

- We're going to learn how to configure **ALL of them**

# In This Course

- We're going to learn how to configure **ALL of them \***
  - Claude Desktop
  - MCP Inspector
  - Postman



# In This Course

- We're going to learn how to configure **ALL of them \***
  - Claude Desktop
  - MCP Inspector
  - Postman
- Demystify the concepts and get a solid grasp of how things work



# Are There Other Clients?

Model Context Protocol Version 2025-06-18 (latest) Search... GitHub ☀️

User Guide Examples Copy page

Introduction

Quickstart >

Concepts >

Examples >

Example Servers

Example Clients

Tutorials >

FAQs

Protocol

Specification

Key Changes

Architecture

Base Protocol >

Client Features >

Roots

Sampling

## Example Clients

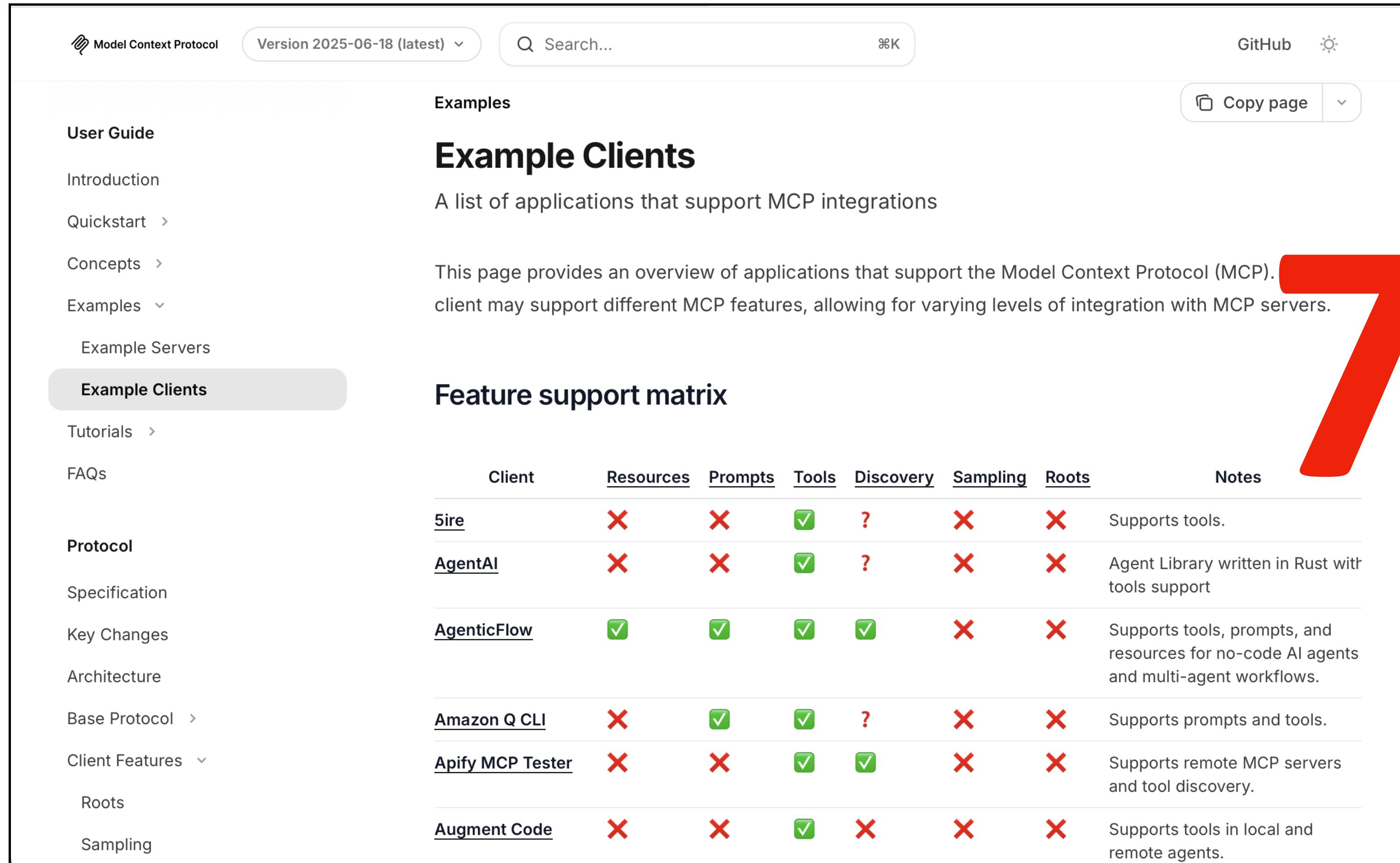
A list of applications that support MCP integrations

This page provides an overview of applications that support the Model Context Protocol (MCP). Each client may support different MCP features, allowing for varying levels of integration with MCP servers.

### Feature support matrix

Client	Resources	Prompts	Tools	Discovery	Sampling	Roots	Notes
<a href="#">5ire</a>	✗	✗	✓	?	✗	✗	Supports tools.
<a href="#">AgentAI</a>	✗	✗	✓	?	✗	✗	Agent Library written in Rust with tools support
<a href="#">AgenticFlow</a>	✓	✓	✓	✓	✗	✗	Supports tools, prompts, and resources for no-code AI agents and multi-agent workflows.
<a href="#">Amazon Q CLI</a>	✗	✓	✓	?	✗	✗	Supports prompts and tools.
<a href="#">Apify MCP Tester</a>	✗	✗	✓	✓	✗	✗	Supports remote MCP servers and tool discovery.
<a href="#">Augment Code</a>	✗	✗	✓	✗	✗	✗	Supports tools in local and remote agents.

# Are There Other Clients?



The screenshot shows the 'Example Clients' page of the Model Context Protocol documentation. The page includes a sidebar with links like 'User Guide', 'Introduction', 'Quickstart', 'Concepts', 'Examples', 'Example Servers', 'Example Clients' (which is highlighted), 'Tutorials', and 'FAQs'. The main content area features a heading 'Example Clients' and a sub-section 'Feature support matrix'. This matrix table compares seven clients against eight features: Resources, Prompts, Tools, Discovery, Sampling, Roots, and Notes. The clients listed are 5ire, AgentAI, AgenticFlow, Amazon Q CLI, Apify MCP Tester, and Augment Code. The notes column provides additional details for each client, such as tool support or specific features like remote servers and tool discovery.

Client	Resources	Prompts	Tools	Discovery	Sampling	Roots	Notes
<a href="#">5ire</a>	✗	✗	✓	?	✗	✗	Supports tools.
<a href="#">AgentAI</a>	✗	✗	✓	?	✗	✗	Agent Library written in Rust with tools support
<a href="#">AgenticFlow</a>	✓	✓	✓	✓	✗	✗	Supports tools, prompts, and resources for no-code AI agents and multi-agent workflows.
<a href="#">Amazon Q CLI</a>	✗	✓	✓	?	✗	✗	Supports prompts and tools.
<a href="#">Apify MCP Tester</a>	✗	✗	✓	✓	✗	✗	Supports remote MCP servers and tool discovery.
<a href="#">Augment Code</a>	✗	✗	✓	✗	✗	✗	Supports tools in local and remote agents.

# What's the Point About MCP?



# Here's the Point About MCP

- It should work like HTTP



# Here's the Point About MCP

- It should work like HTTP
- Take any client that you want



# Here's the Point About MCP

- It should work like HTTP
- Take any client that you want
- The client should not have any knowledge of server



# Here's the Point About MCP

- It should work like HTTP
- Take any client that you want
- The client should not have any knowledge of server
- Access the MCP server that you want to perform the thing that you want to do



# Can Claude Desktop Get the Weather?

## DEMO 2

# Can Claude Desktop Get the Weather?

## DEMO 2



## Claude Desktop

# Can Claude Desktop Get the Weather?

## DEMO 2

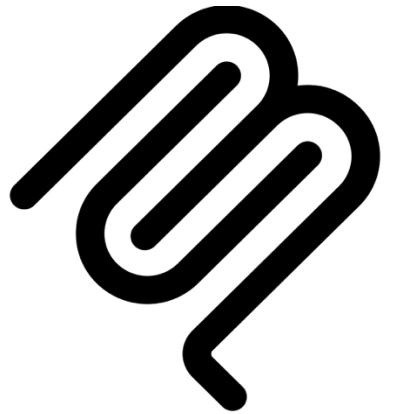


### PROMPT:

*What's the weather in Dallas?*



Claude Desktop



# Questions?



# 7 min Break

## PART 2

### Getting Started with MCP Servers



# Audience Poll Question #3



# Audience Poll Question #3

**Which is your preferred programming language? (Single response)**

- Python
- Java
- Node.js
- C#
- Kotlin
- Rust
- Swift
- Other



# What have we accomplished so far?

- We understand **why** the industry has overwhelming support for MCP
- We understand that it's a **Client/ Server** architecture like HTTP
- We saw an MCP client to use a basic MCP Server



# What can be improved upon?

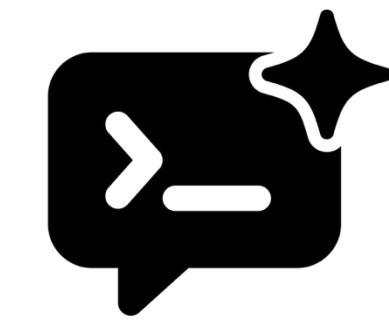


- We understand how clients work, but we don't understand **how servers work yet**
- We haven't **covered the code** necessary to build MCP servers
- Let's understand some new terms first

# What are the Basic Terms and Terminology?



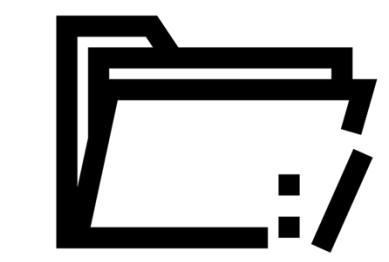
- Message Protocol



- Prompts



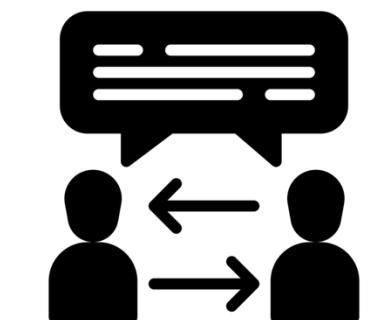
- Transports



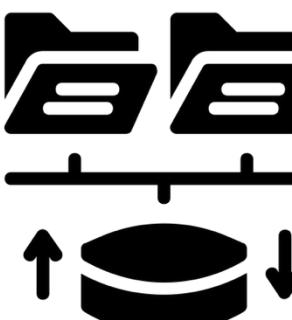
- Roots



- Tools



- Sampling



- Resources



# What are the Basic Terms and Terminology?



- Message Protocol



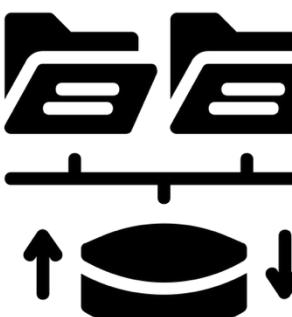
- Prompts



- Transports



- Tools



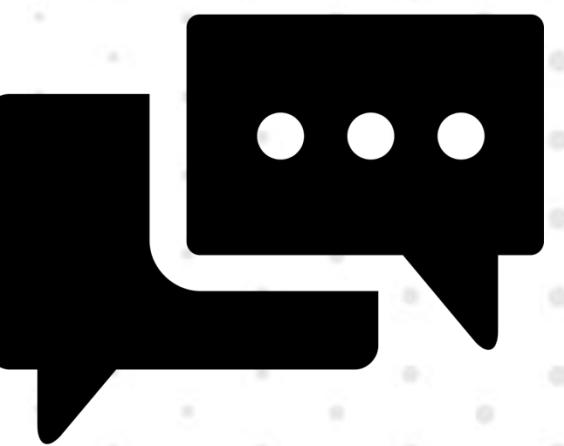
- Resources

- **Roots**

- **Sampling**

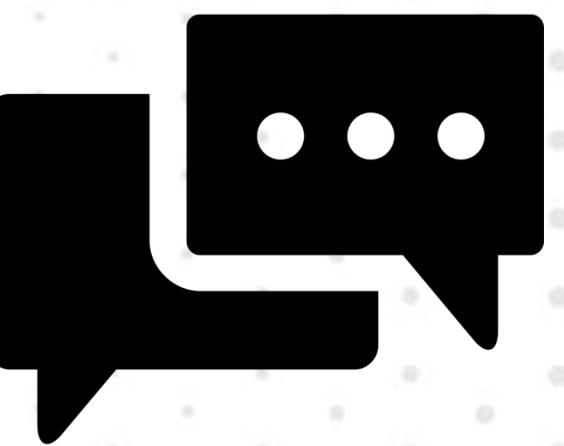


# Message Protocol



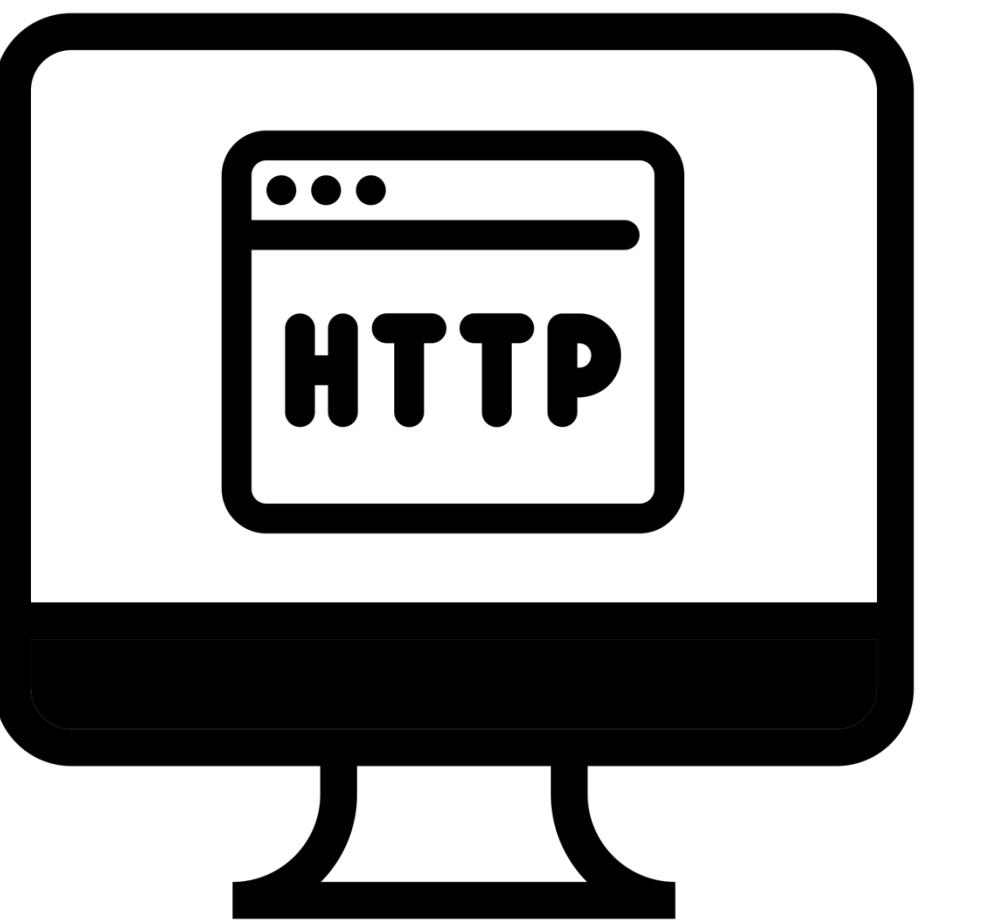
# Message Protocol

- Remember how HTTP works?

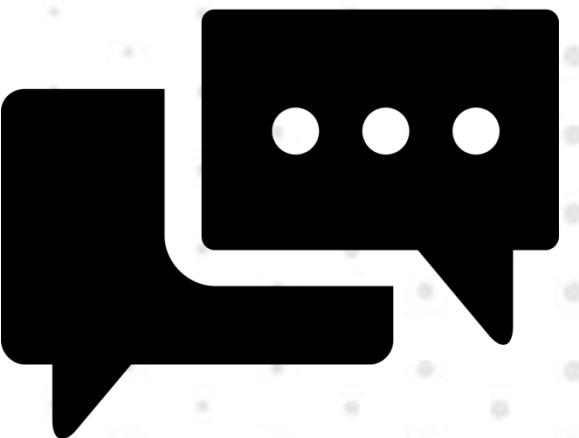


# Message Protocol

- Remember how HTTP works?

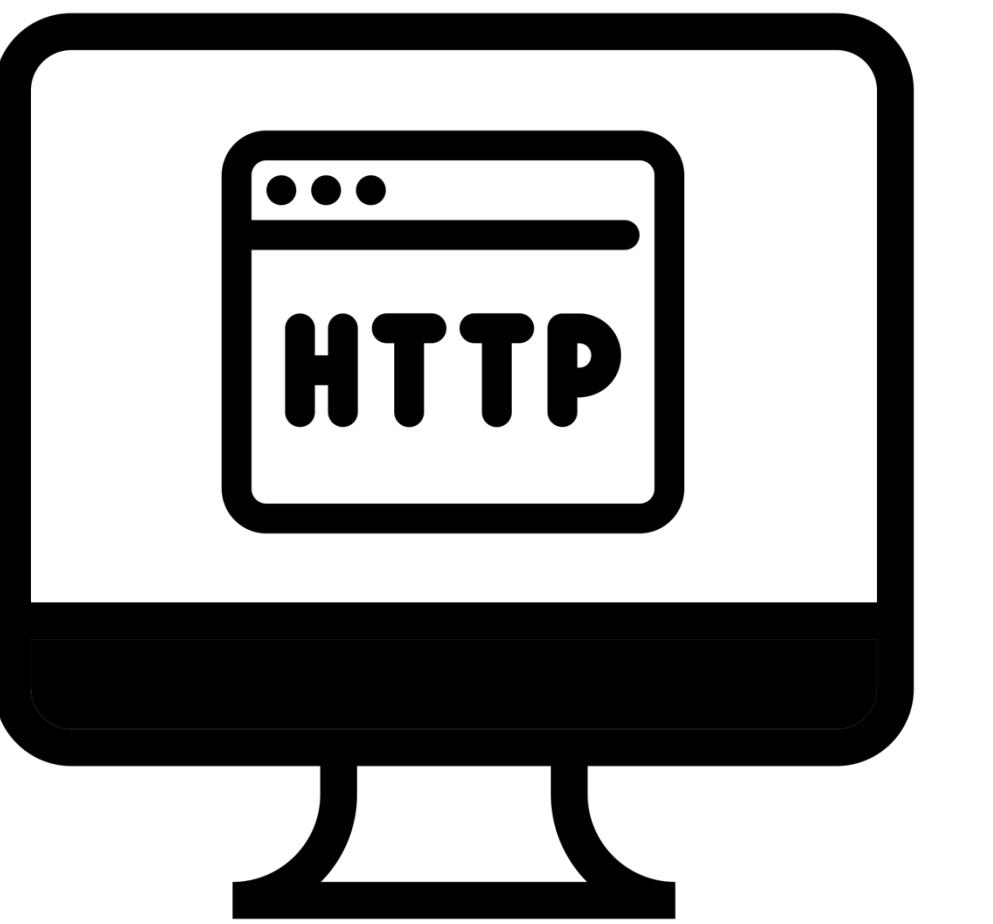


Web browser

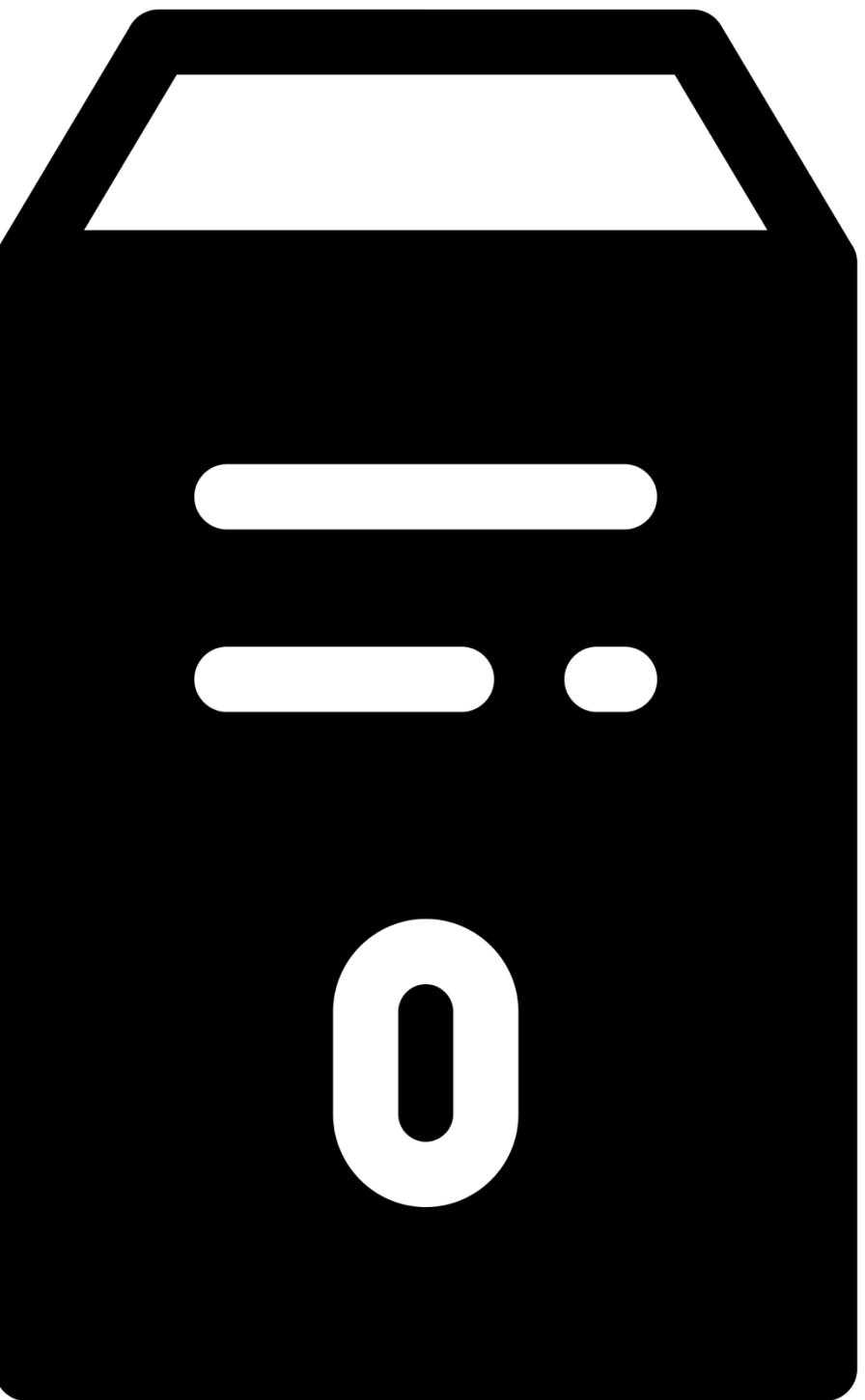


# Message Protocol

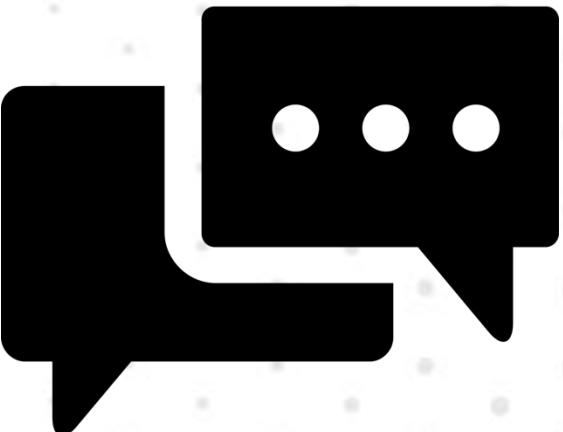
- Remember how HTTP works?



Web browser

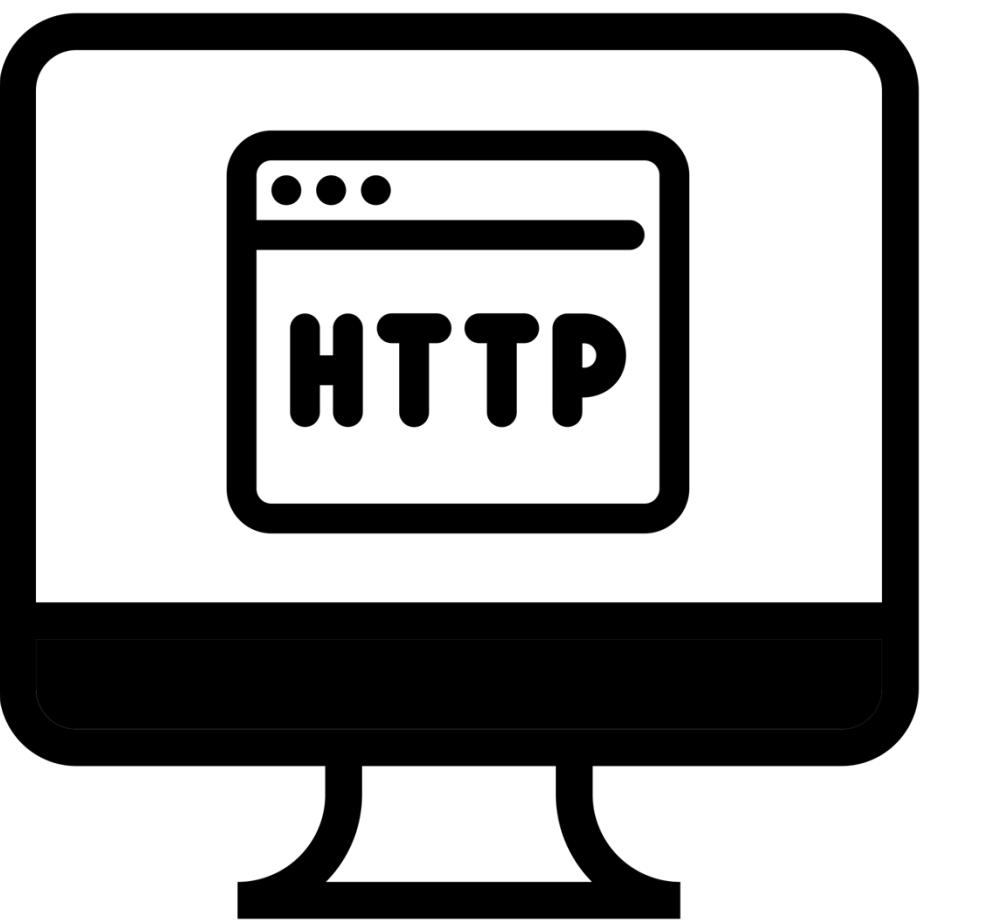


Web server

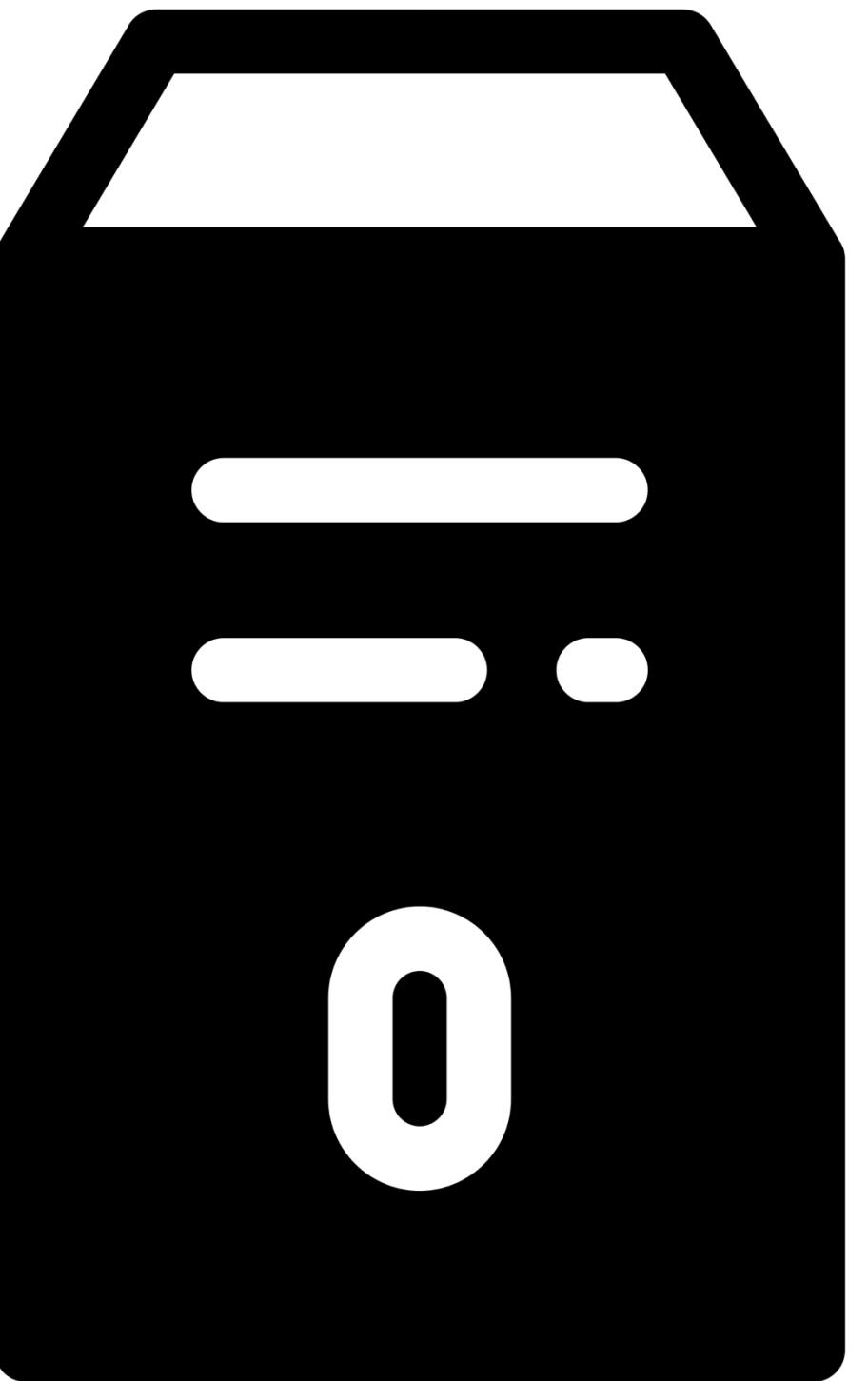
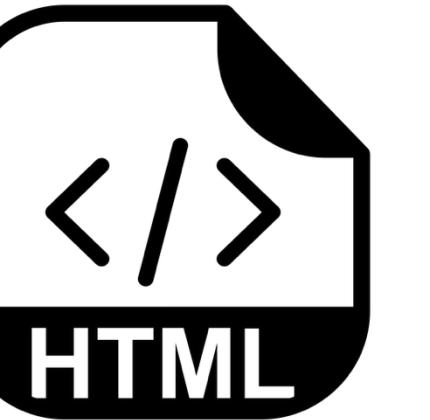


# Message Protocol

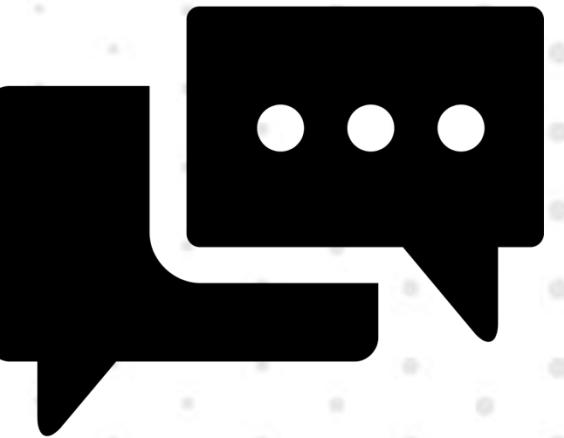
- Remember how HTTP works?



Web browser

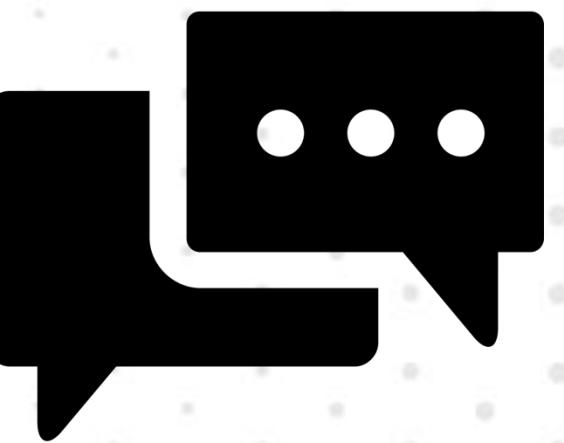


Web server



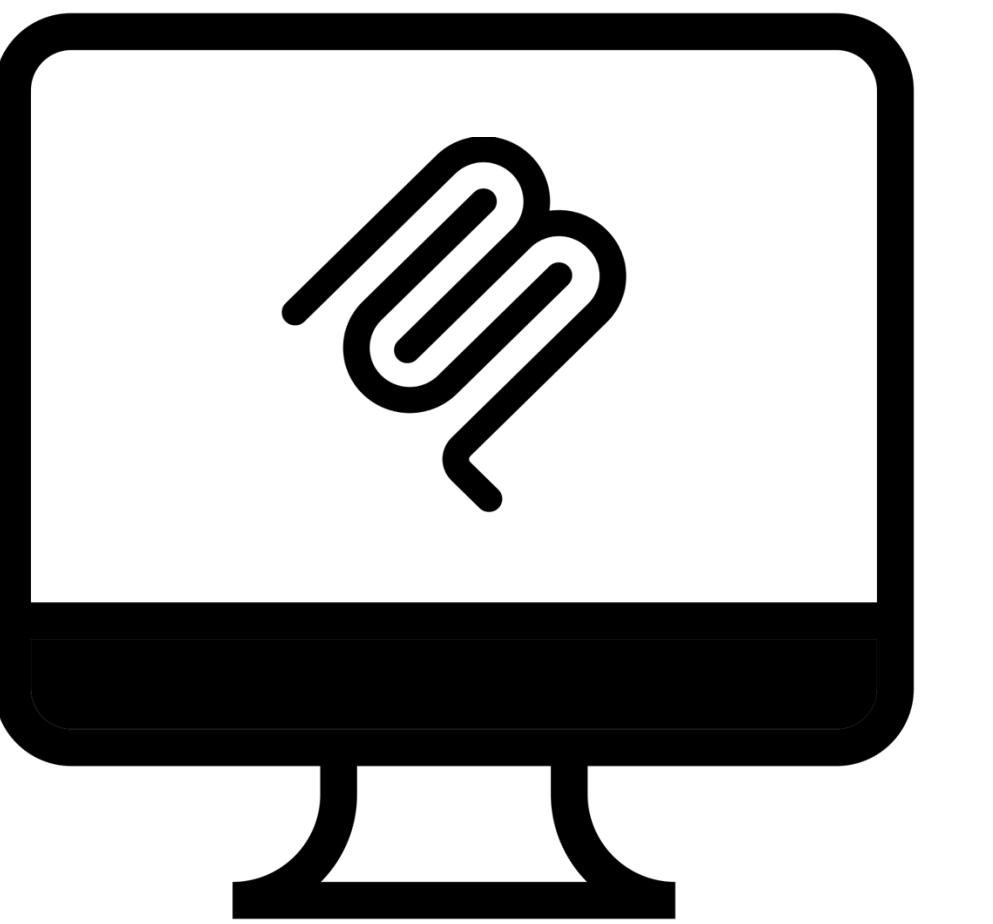
# Message Protocol

- MCP is similar

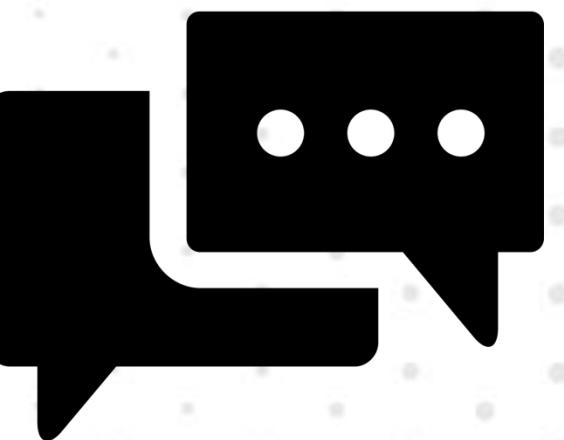


# Message Protocol

- MCP is similar

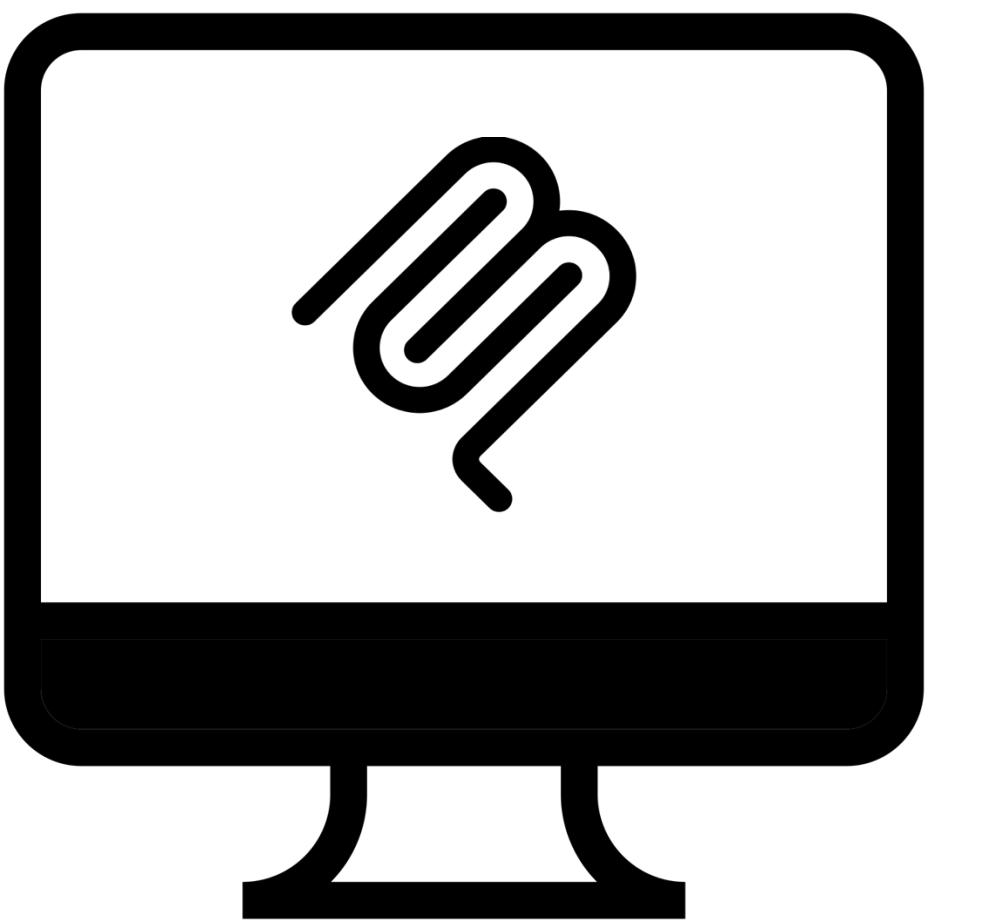


MCP Client

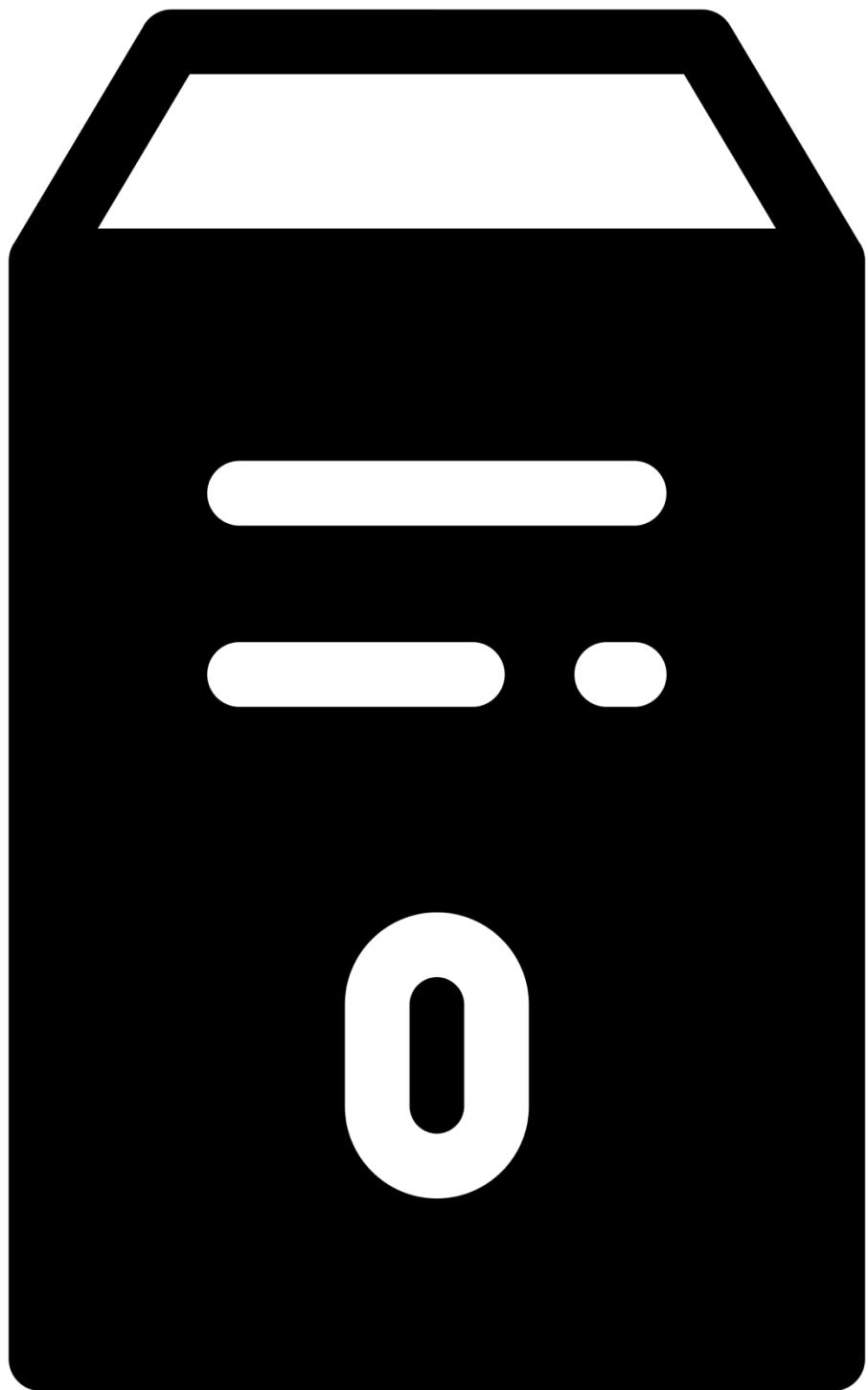


# Message Protocol

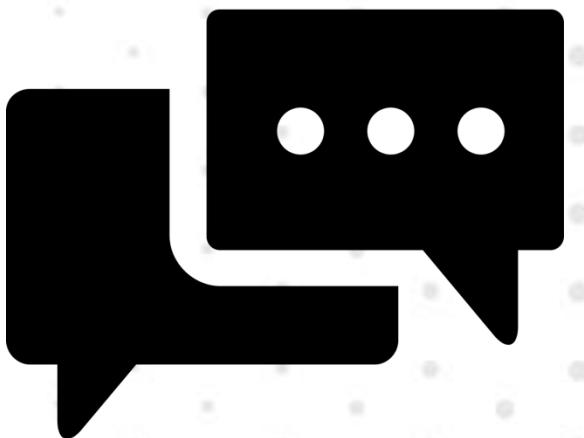
- MCP is similar



MCP Client

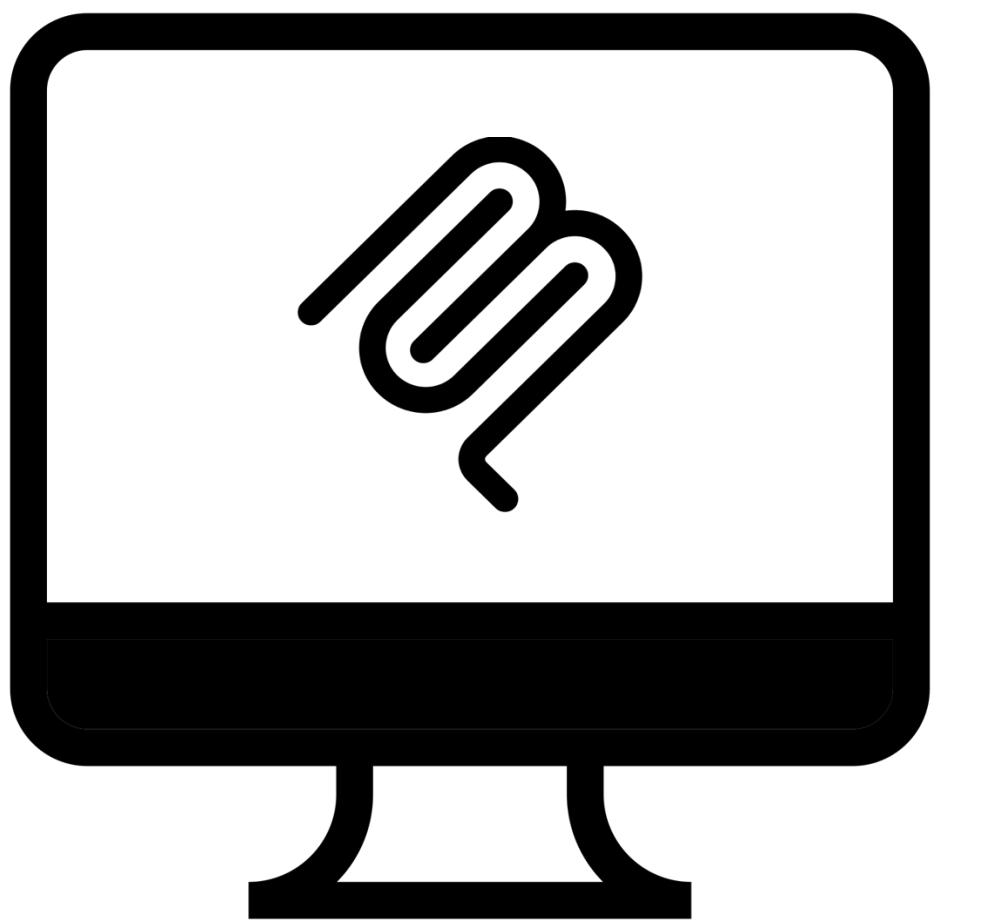


MCP Server

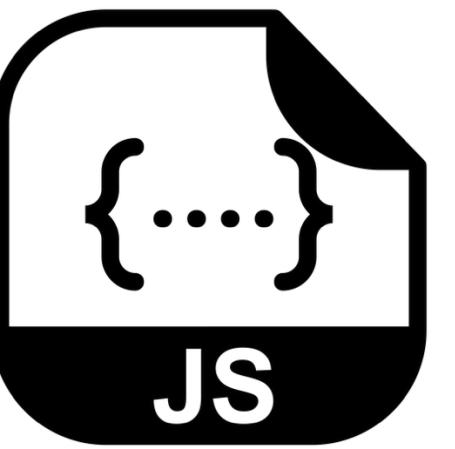


# Message Protocol

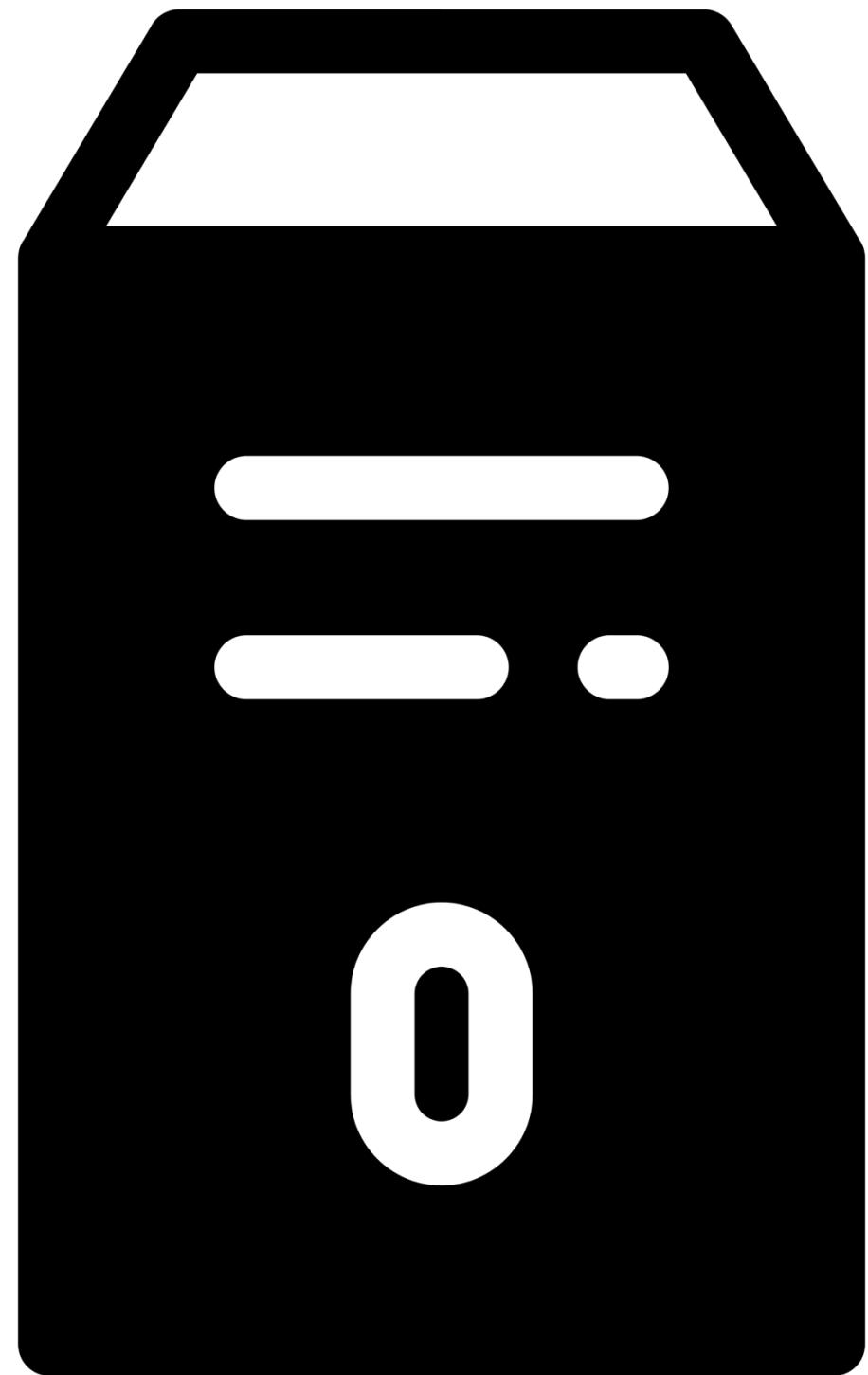
- MCP is similar



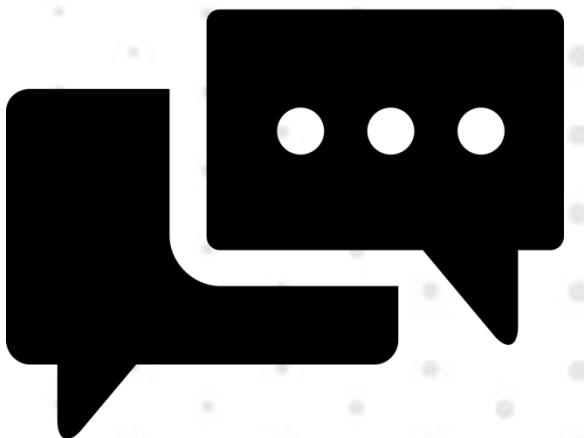
MCP Client



JSON RPC

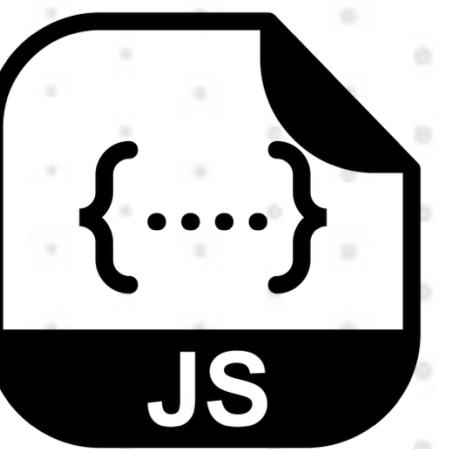


MCP Server

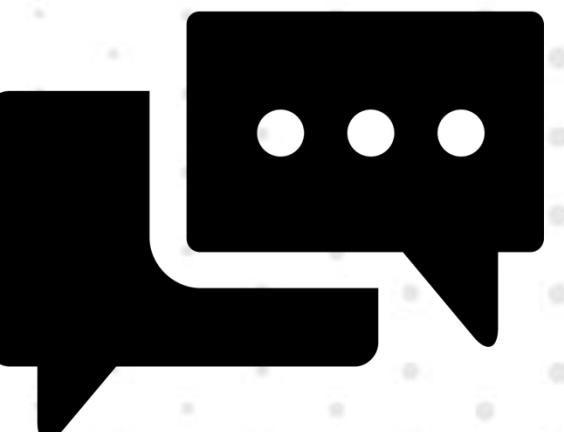


# Message Protocol: JSON-RPC

- For example
- The Client asks the Server,  
**“What tools do you support?”**
- What does that look like?

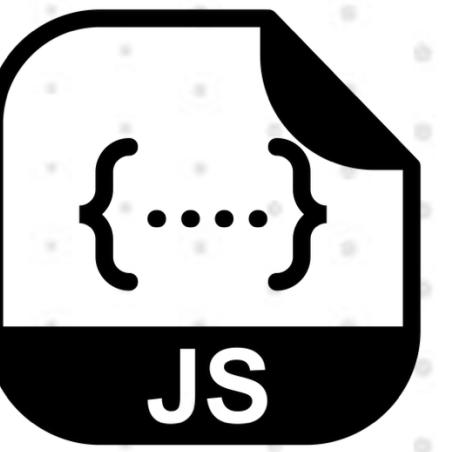


JSON RPC

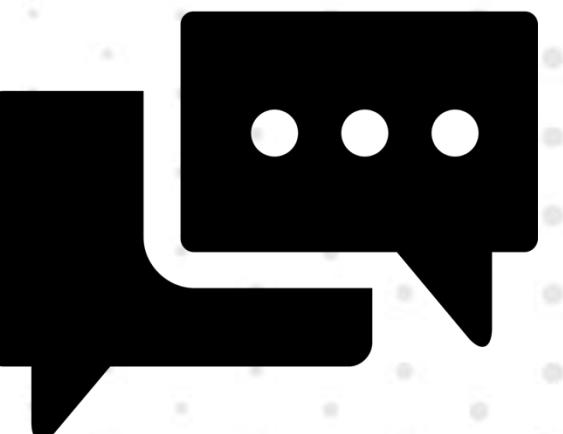


# JSON-RPC: Request

```
{  
  "method": "tools/list",  
  "params": {}  
}
```

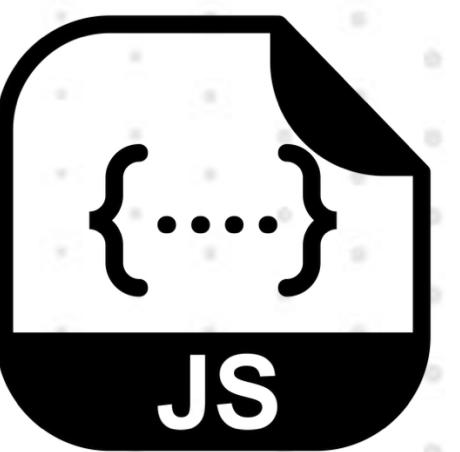


JSON RPC

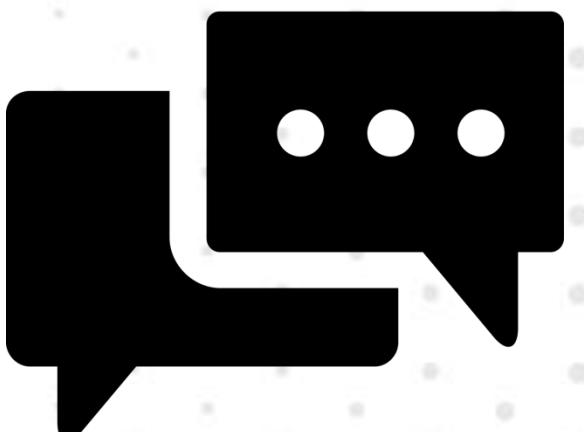


# JSON-RPC: Response

```
{  
  "tools": [  
    {  
      "name": "get_weather",  
      "description": "Get current weather information for a specified city. Returns temperature data for the requested location.",  
      "inputSchema": {  
        "type": "object",  
        "properties": {  
          "city": {  
            "type": "string",  
            "description": "Name of the city to get weather for (e.g., 'New York', 'London', 'Tokyo')"  
          }  
        },  
        "required": [  
          "city"  
        ]  
      }  
    }  
  ]  
}
```



JSON RPC



# Transports

20

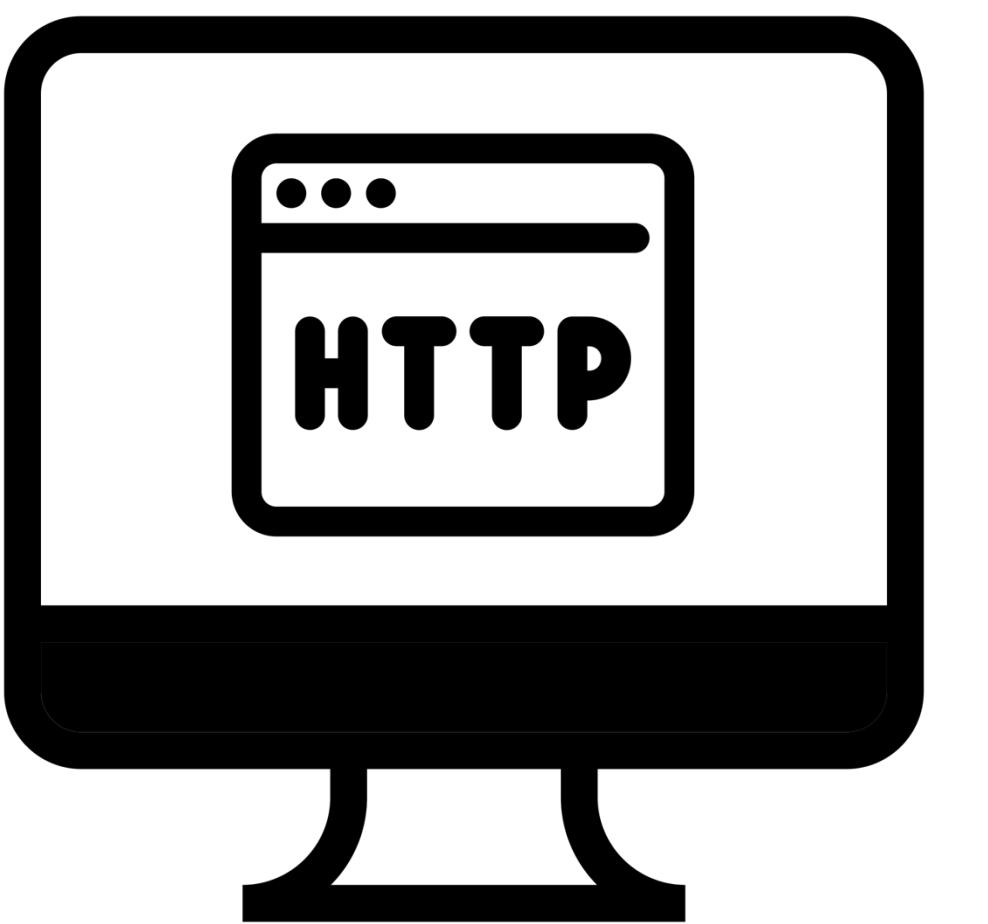
# Transports

- Remember how HTTP works?

2.

# Transports

- Remember how HTTP works?

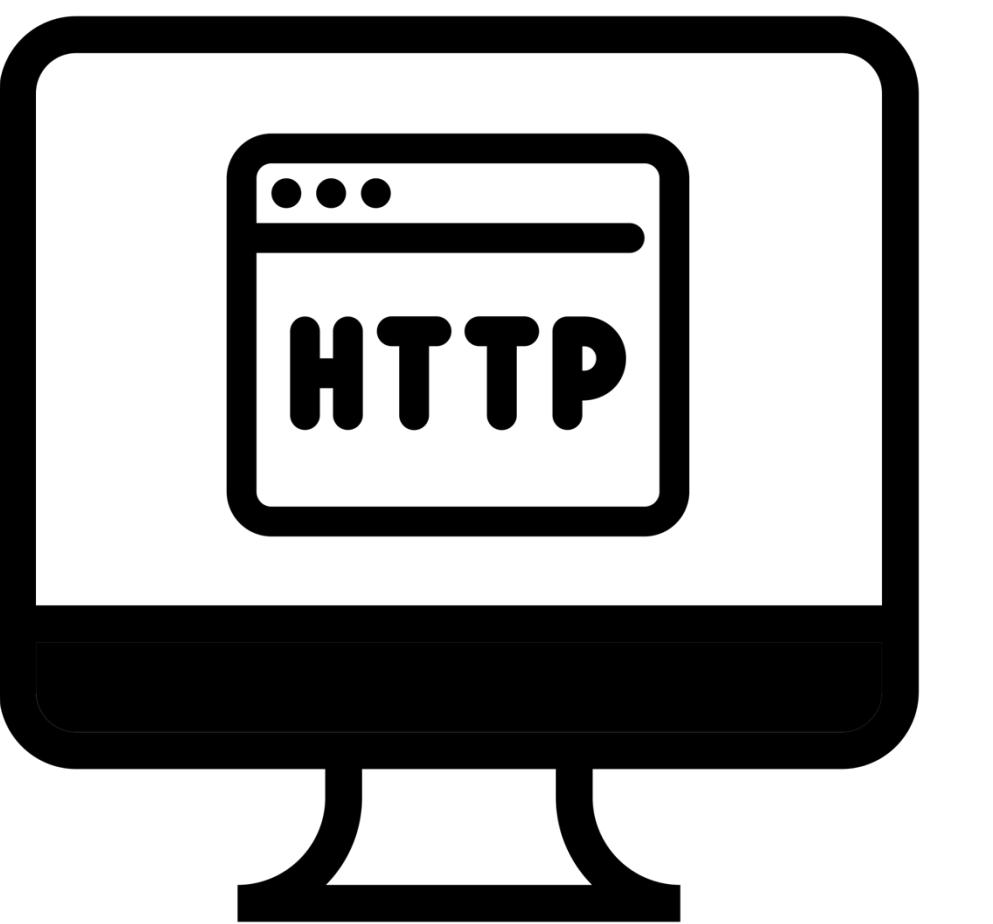


Web browser

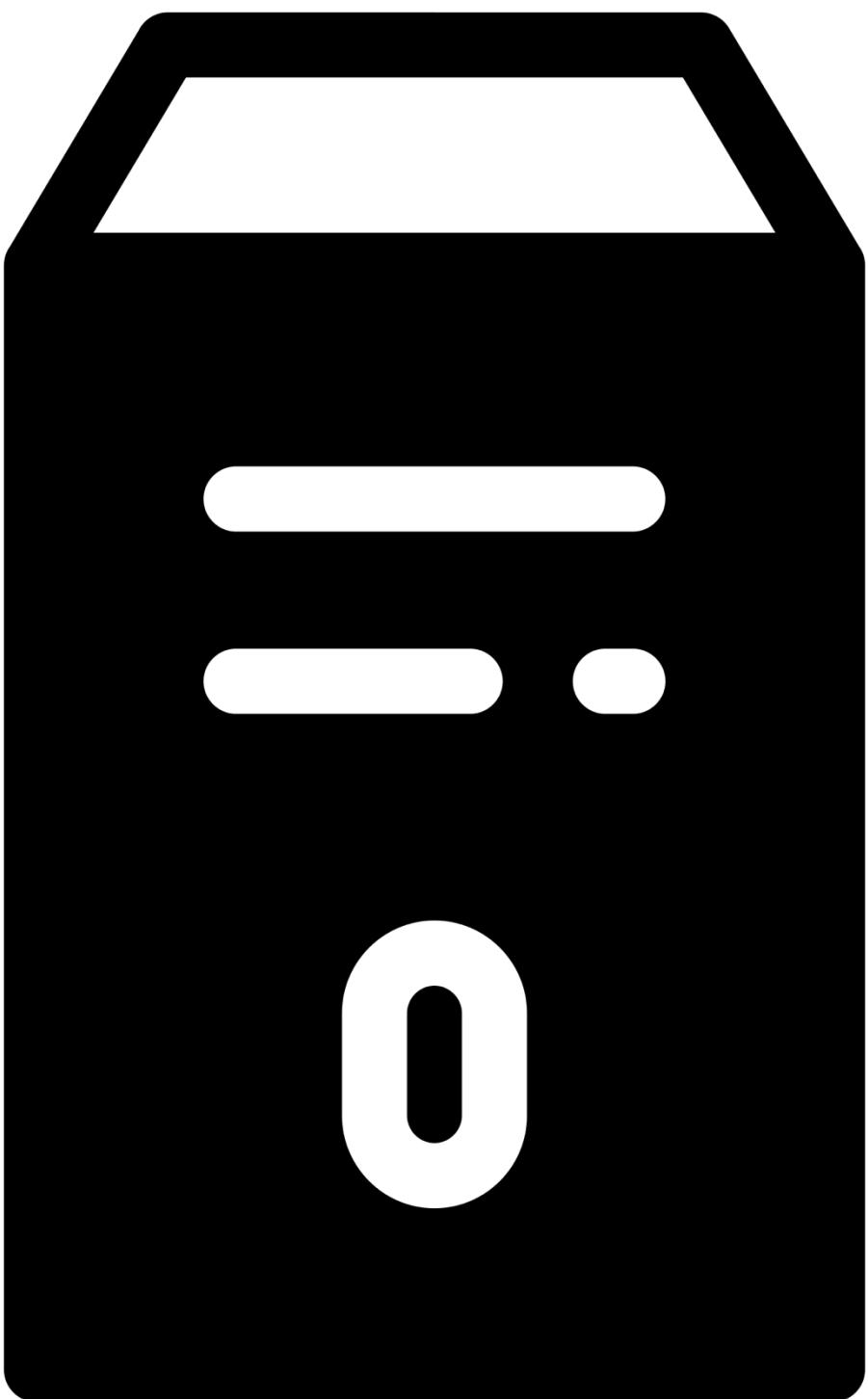
2.

# Transports

- Remember how HTTP works?



Web browser

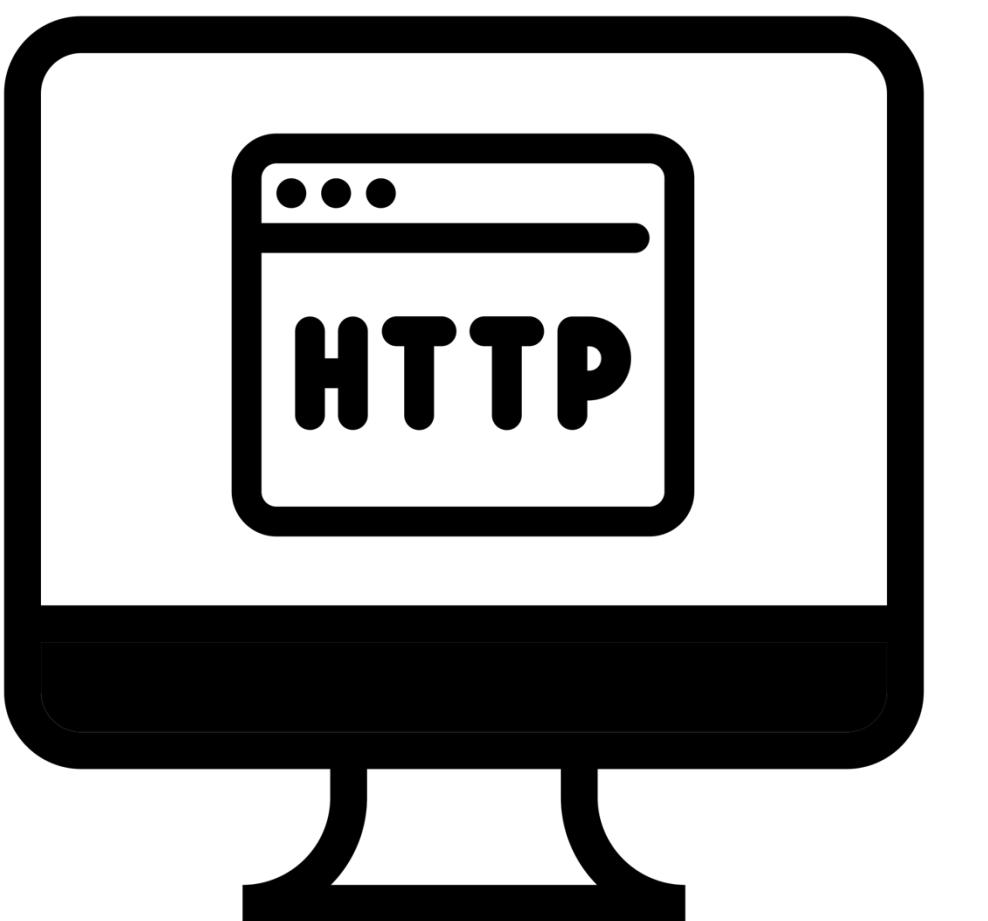


Web server

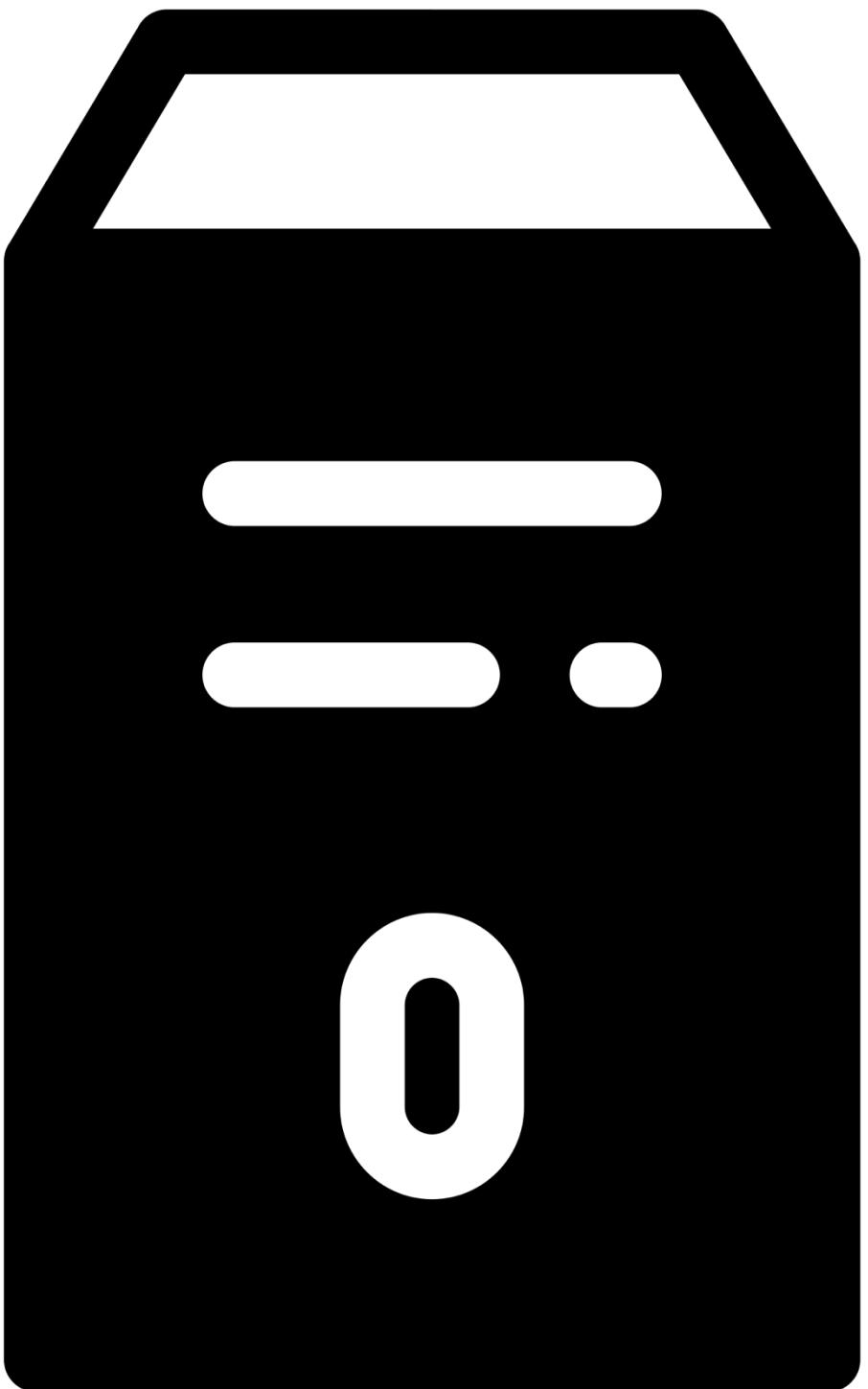


# Transports

- Remember how HTTP works?



Web browser

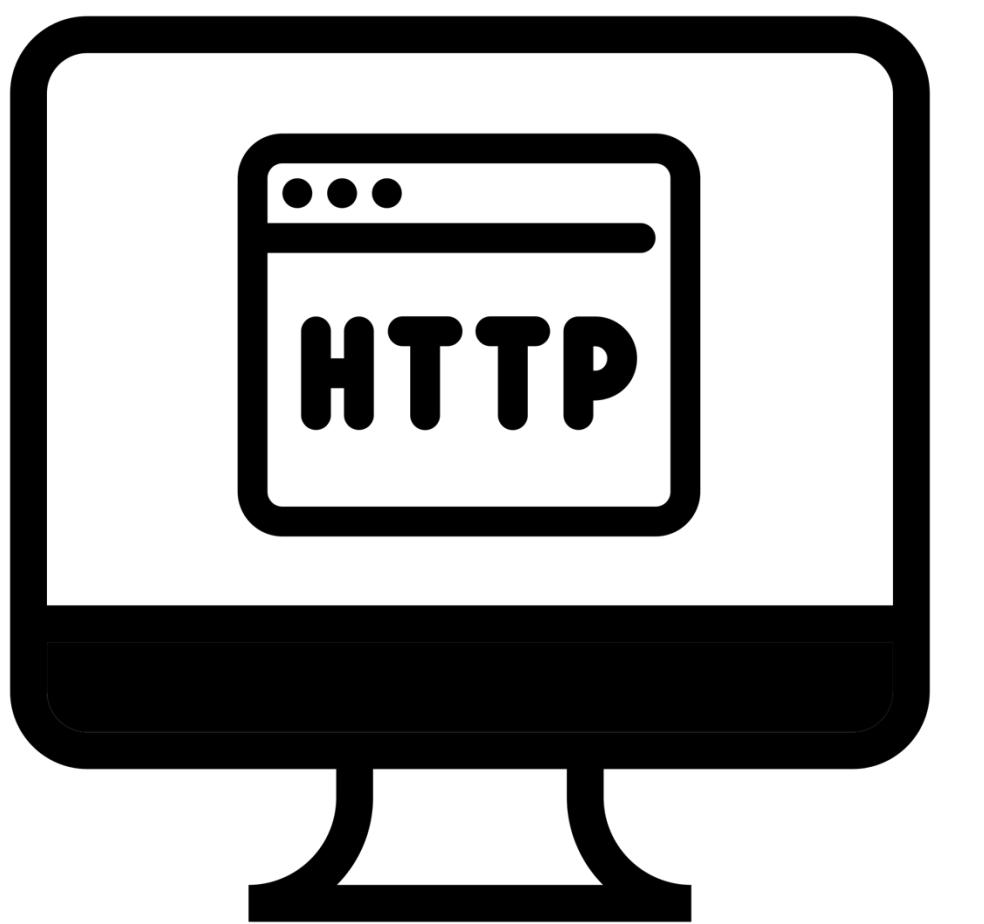


Web server



# Transports

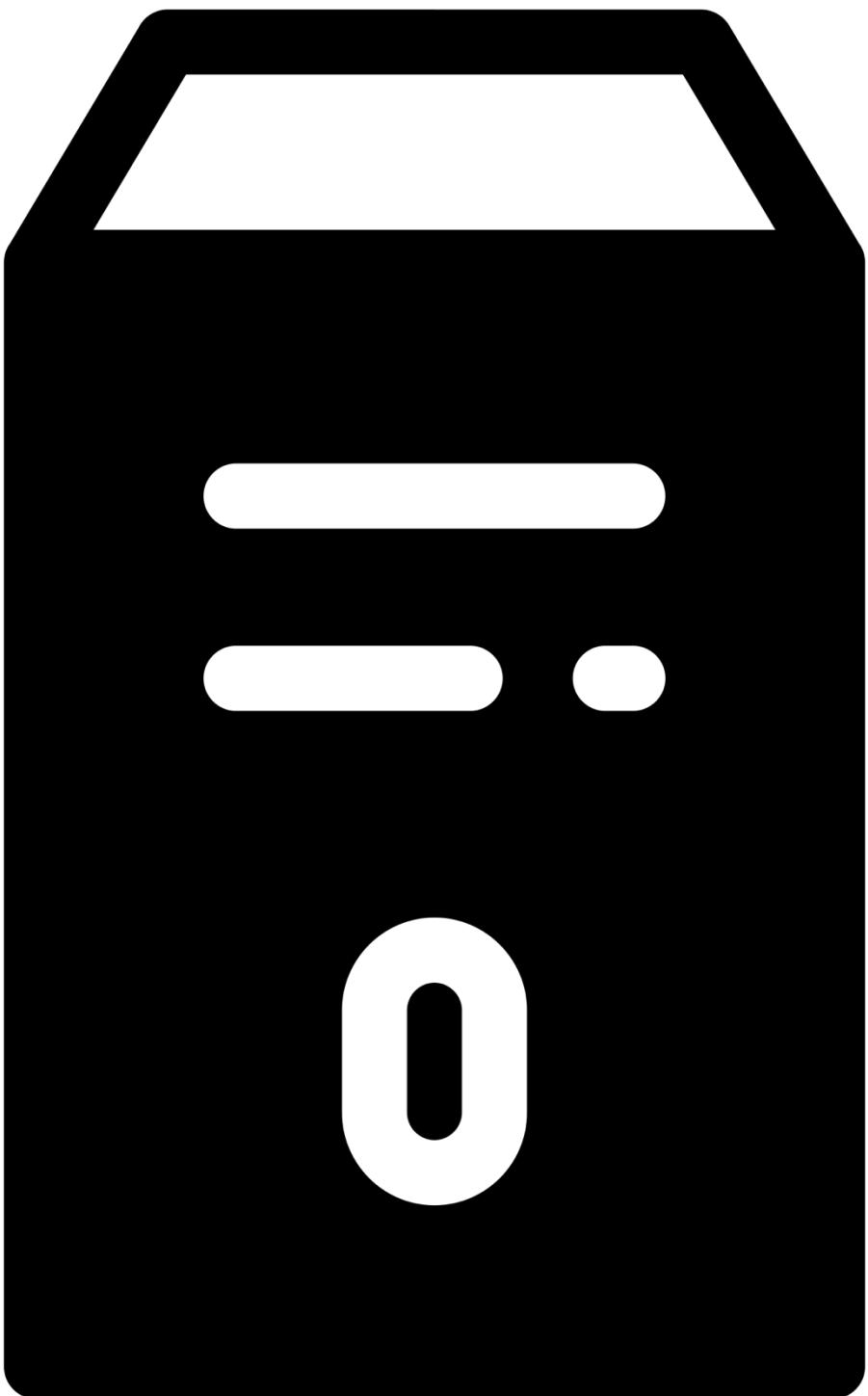
- Remember how HTTP works?



Web browser



TCP or UDP



Web server



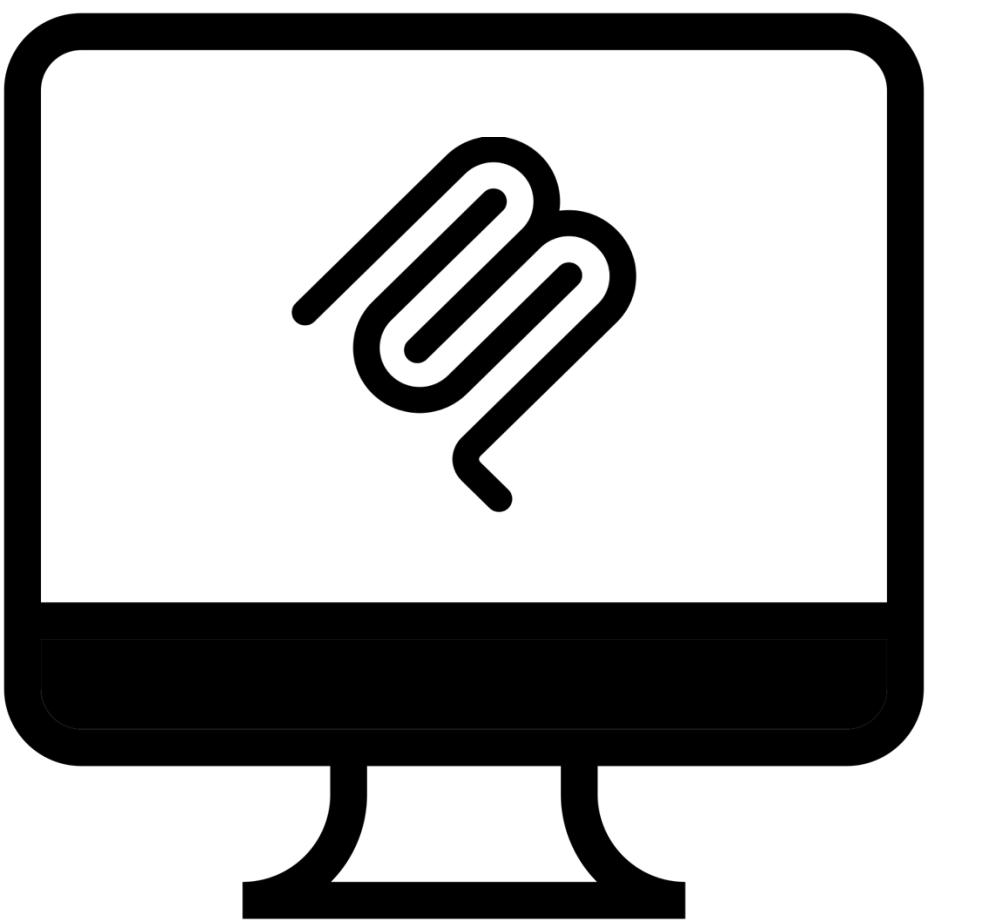
# Transports

- MCP is similar

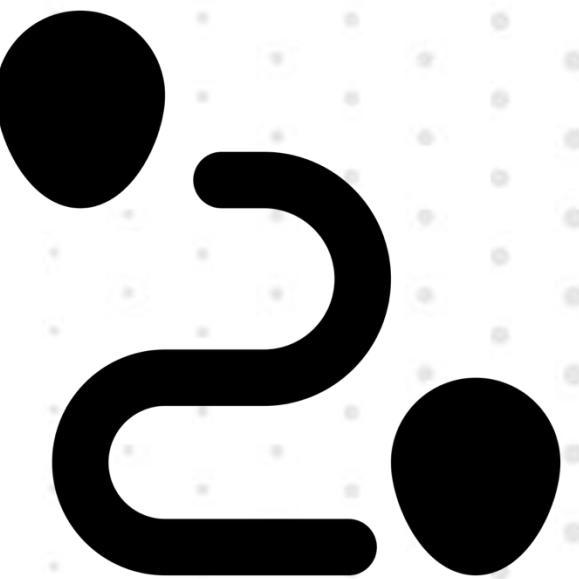
2.

# Transports

- MCP is similar

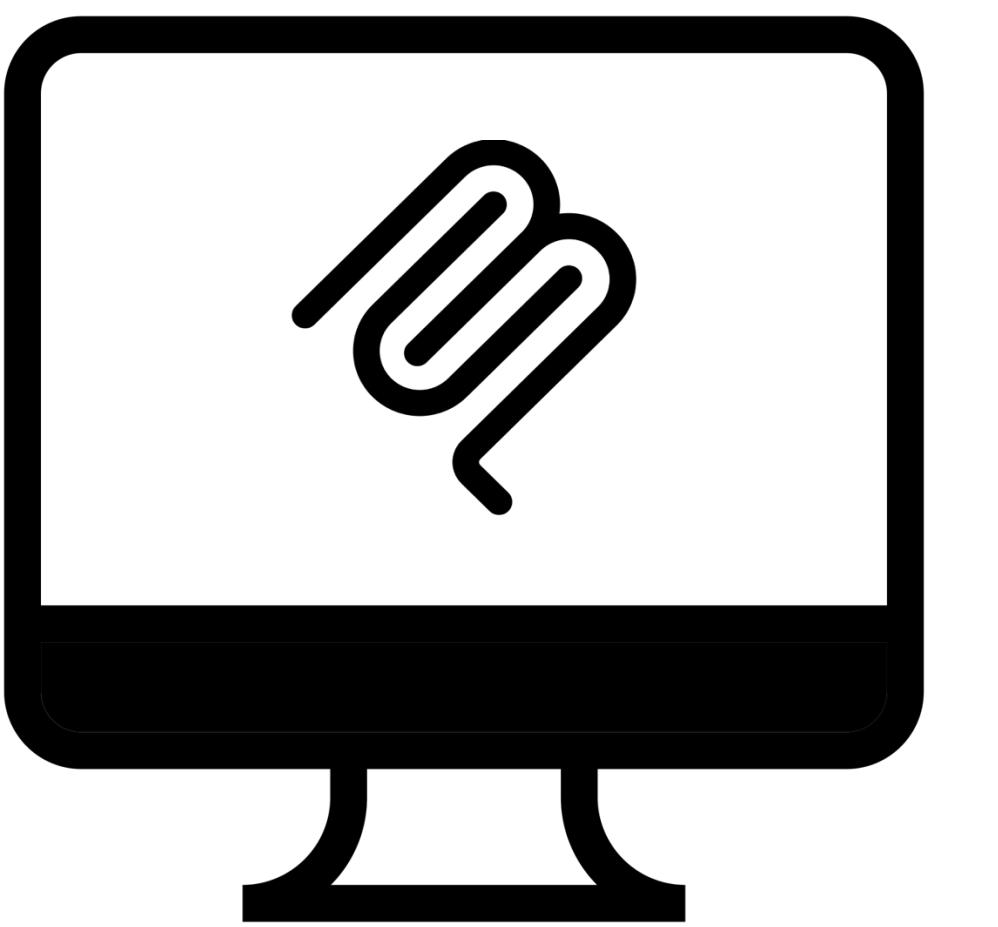


MCP Client

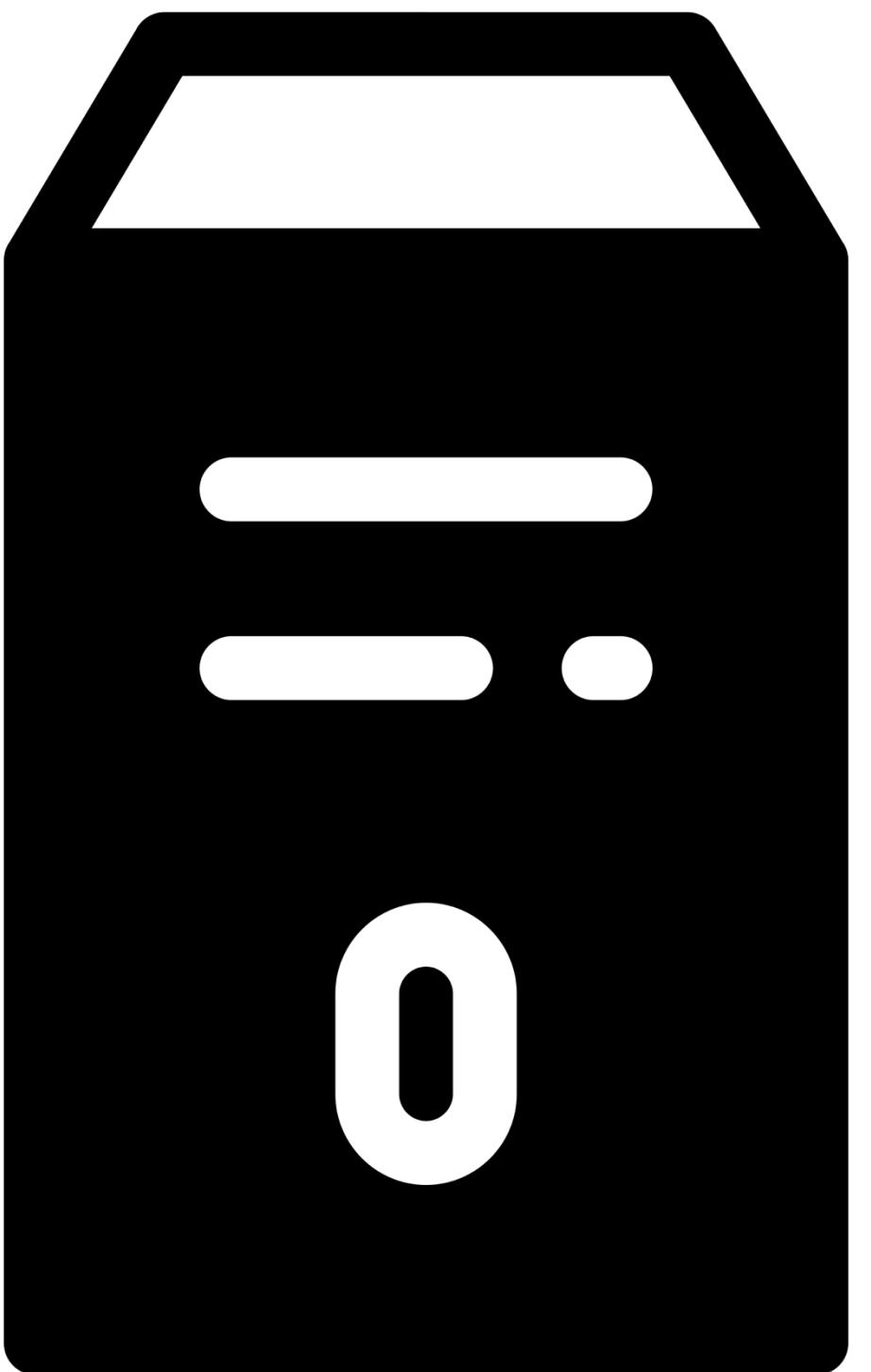


# Transports

- MCP is similar



MCP Client

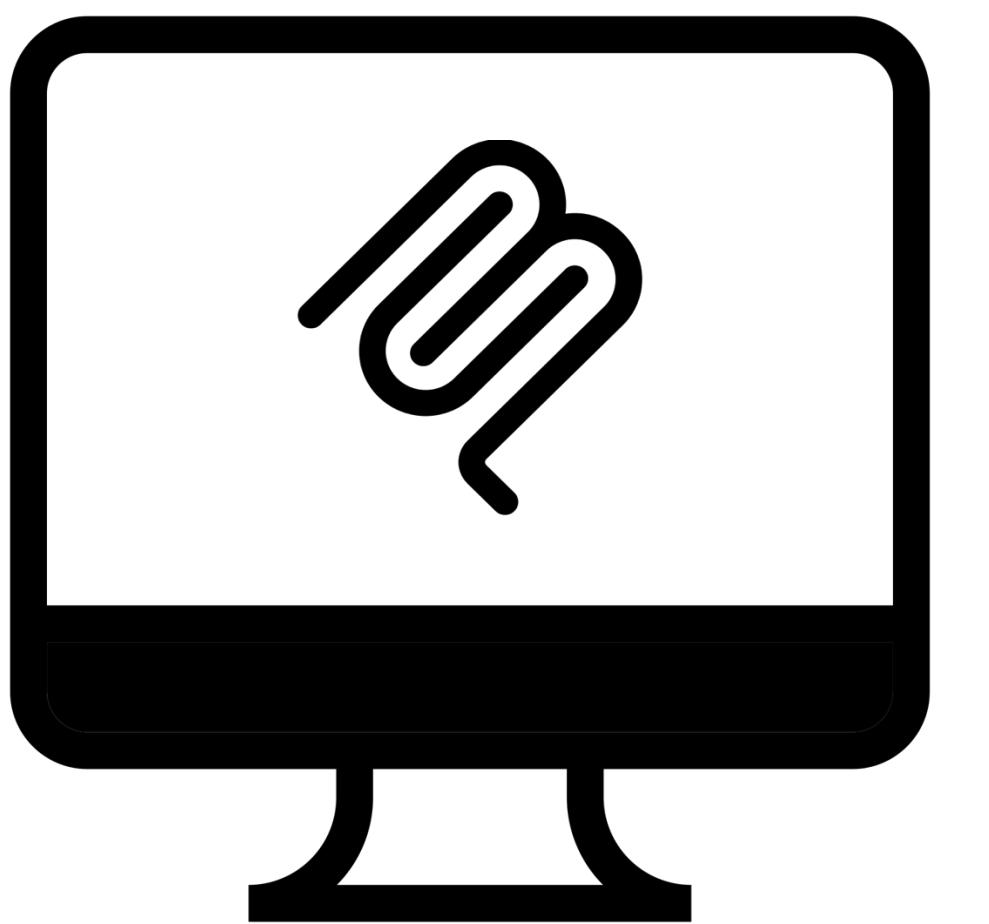


MCP Server

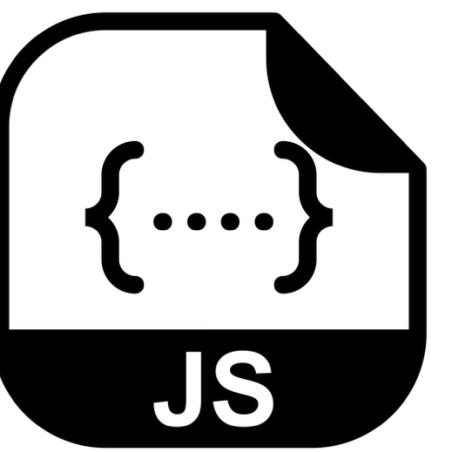


# Transports

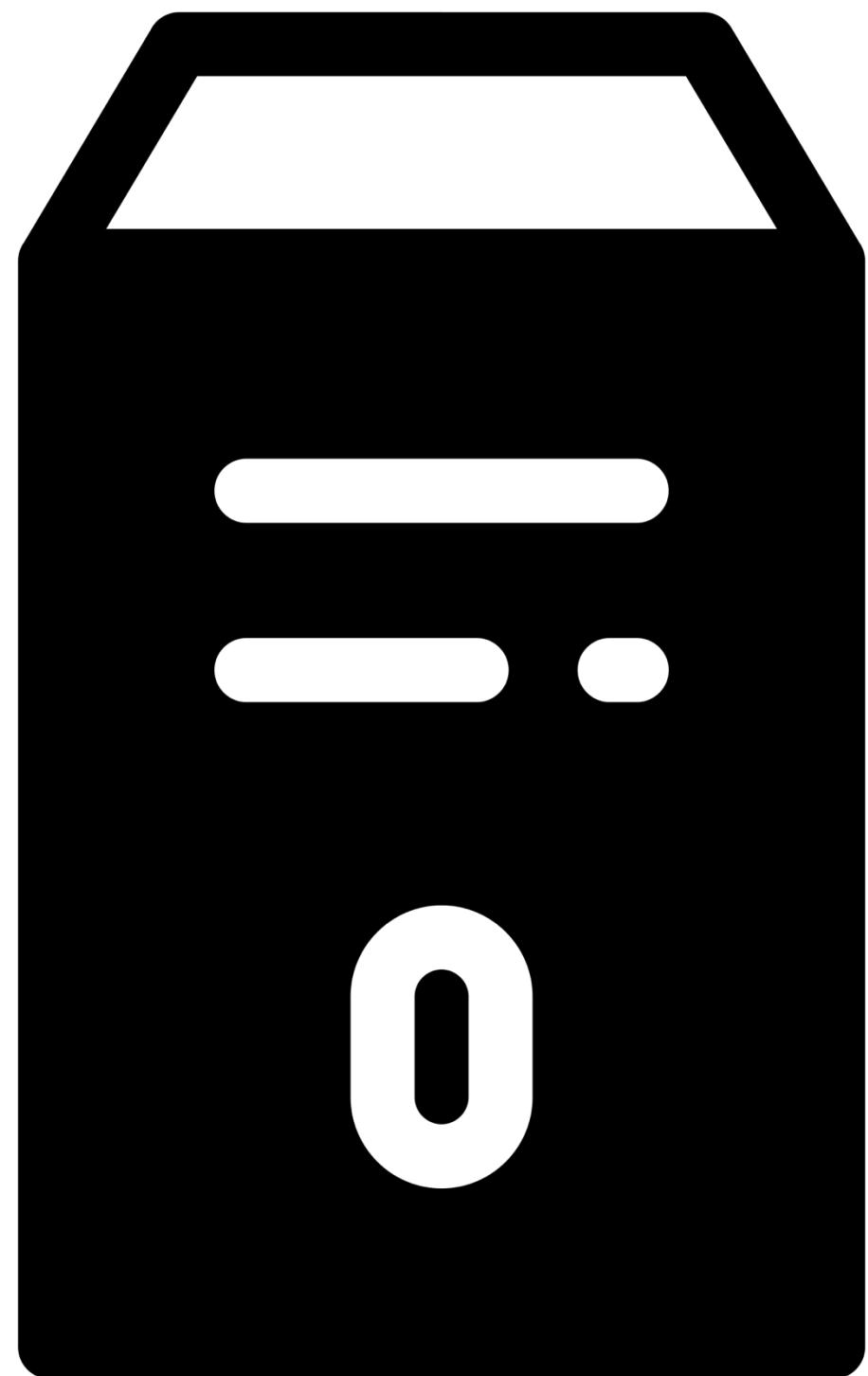
- MCP is similar



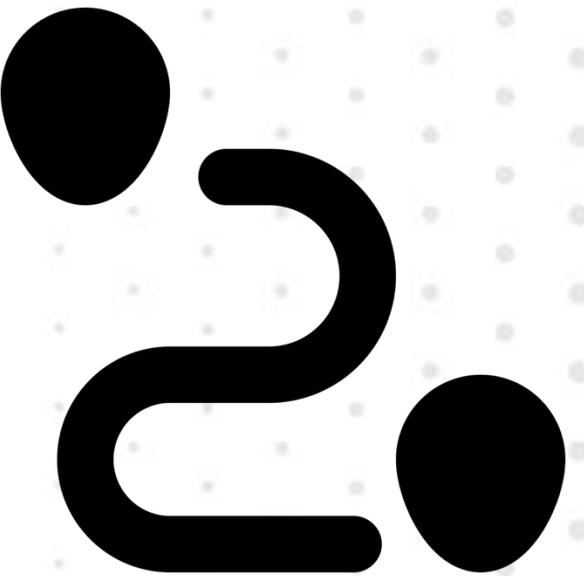
MCP Client



JSON RPC

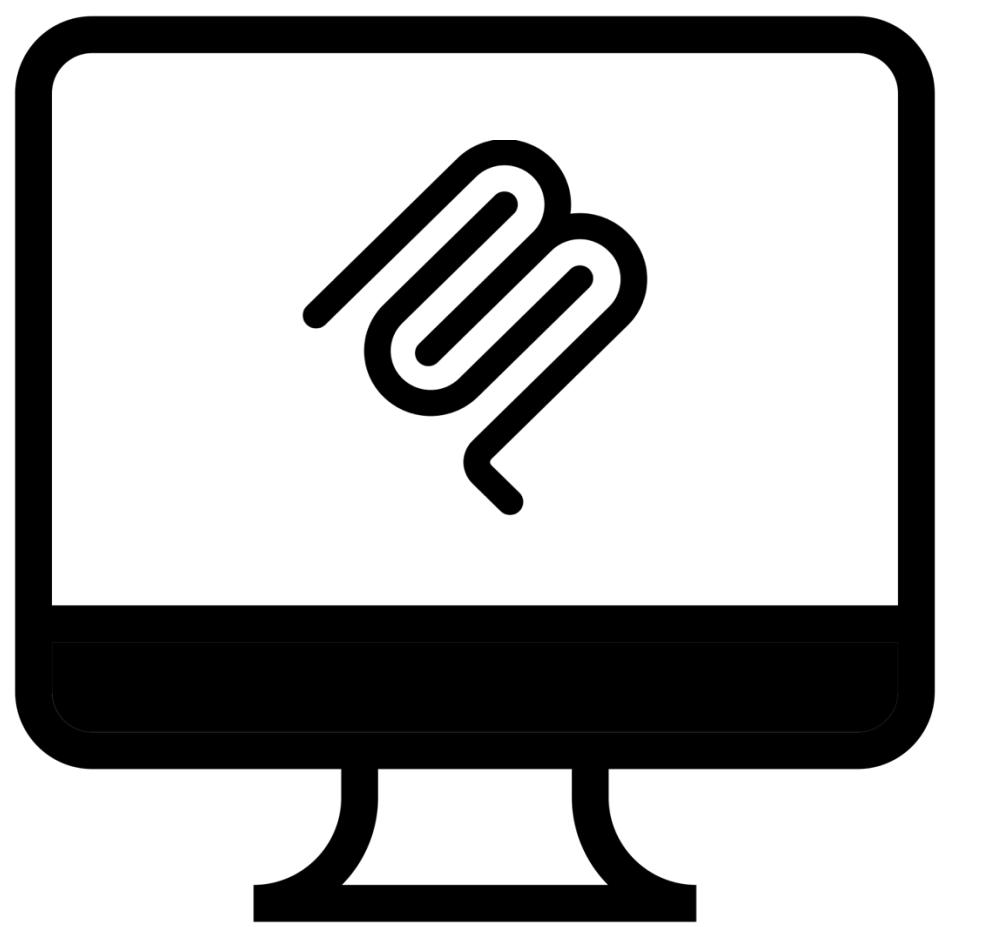


MCP Server

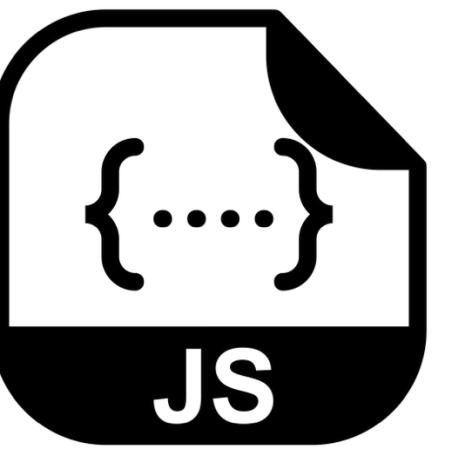


# Transports

- MCP is similar

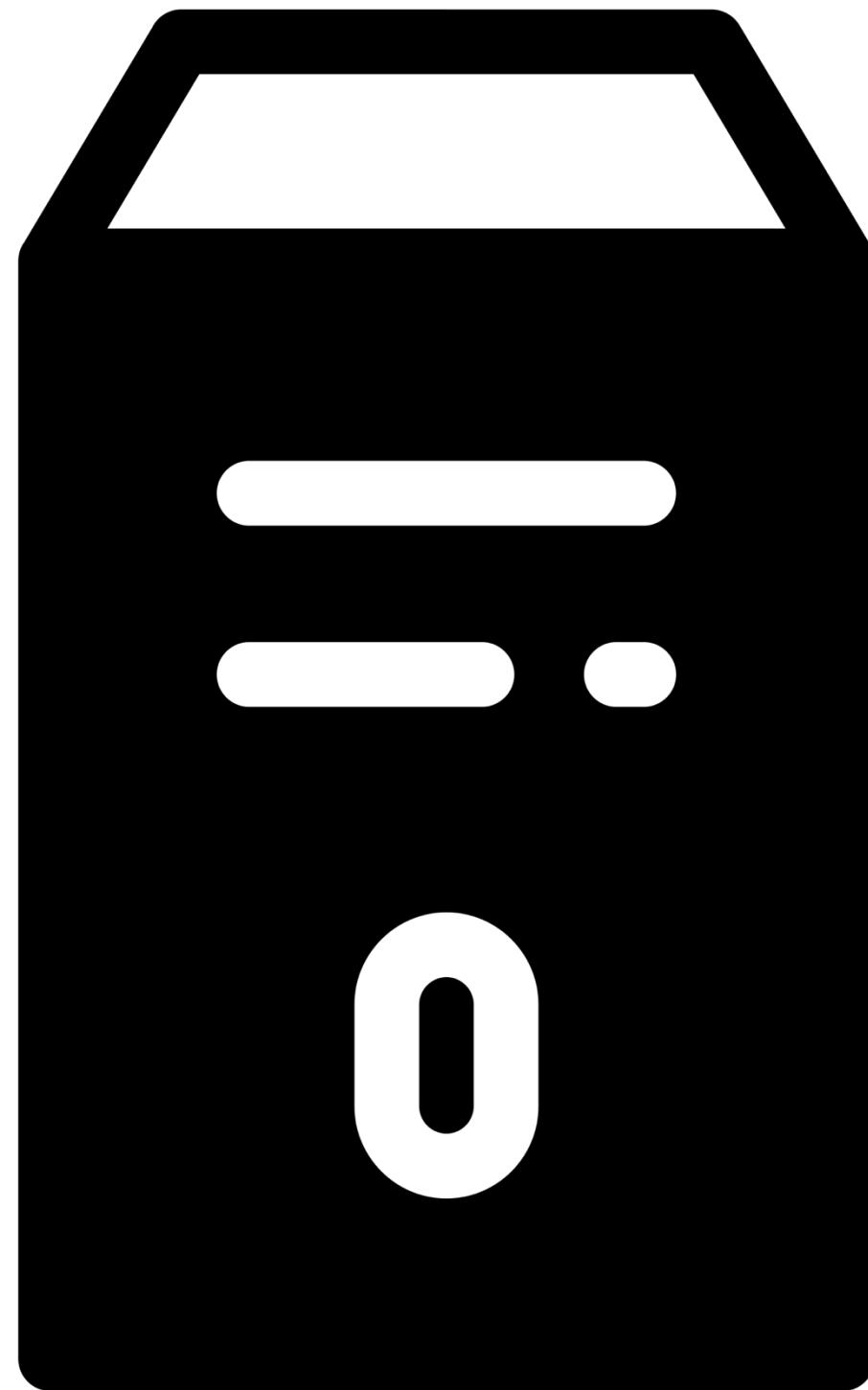


MCP Client



JSON RPC

Standard Input Output  
(STDIO)  
or  
Streamable HTTP



MCP Server



# Wait, wait, wait

- STUDIO is a Transport?



2.

# STDIO is a Transport?

2

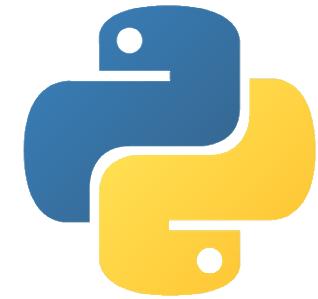
# STUDIO is a Transport?

- Do you mean THIS Standard Output?

2.

# STDIO is a Transport?

- Do you mean THIS Standard Output?

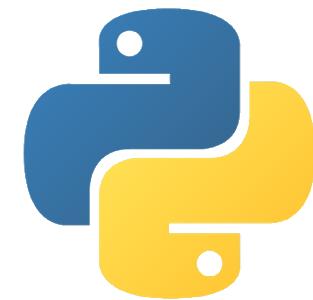


- `print("Checking the weather...")`

2.

# STDIO is a Transport?

- Do you mean THIS Standard Output?



- `print("Checking the weather...")`



- `System.out.println("Checking the weather...");`

2

# STUDIO is a Transport?

- Do you mean THIS Standard Output?



- `print("Checking the weather...")`



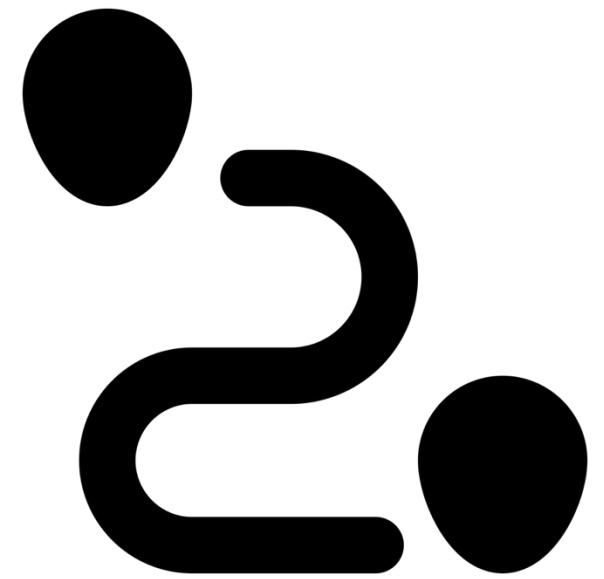
- `System.out.println("Checking the weather...");`



- `console.log("Checking the weather...");`

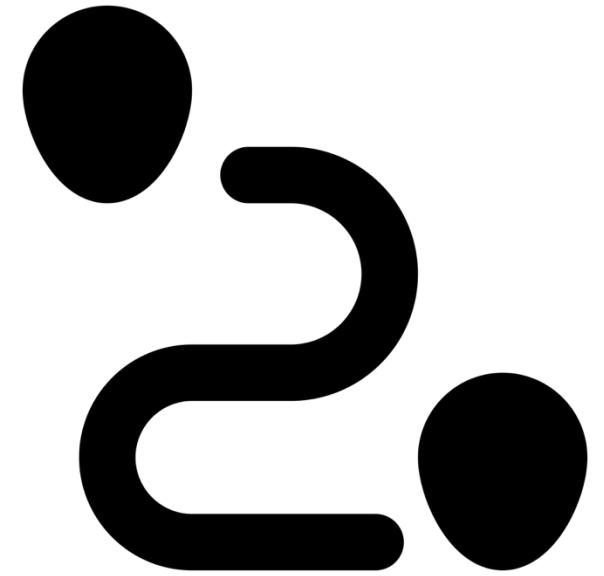


# So What Exactly is a Transport?



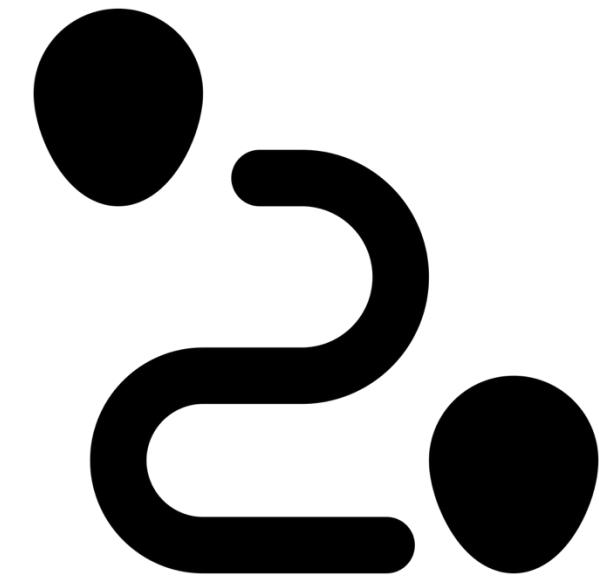
- A Transport is literally whatever you use to get from Point A to Point B

# So What Exactly is a Transport?



- A Transport is literally whatever you use to get from Point A to Point B
- Both A and B need common access to the transport

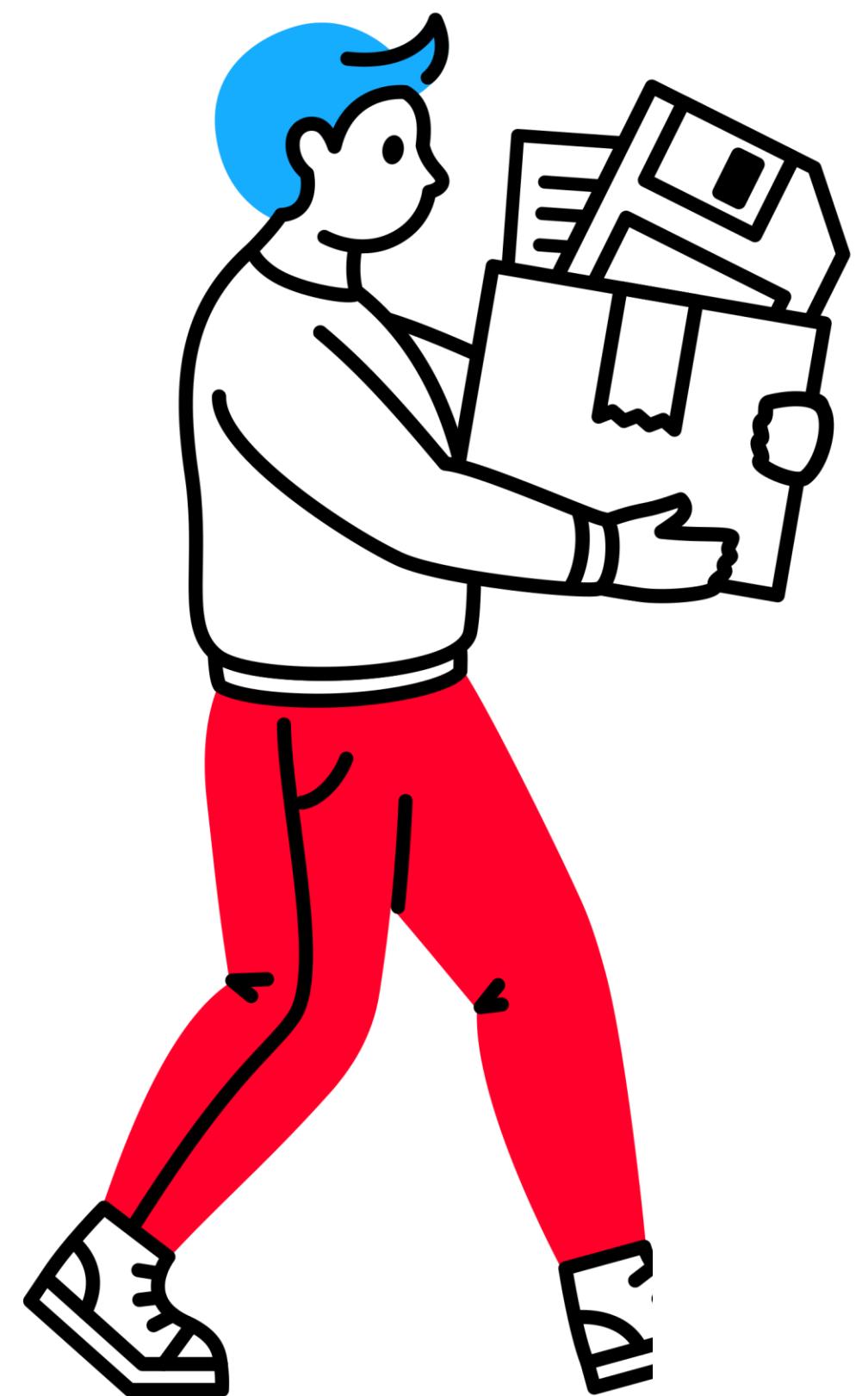
# So What Exactly is a Transport?



- A Transport is literally whatever you use to get from Point A to Point B
- Both A and B need common access to the transport
- A transport can be anything

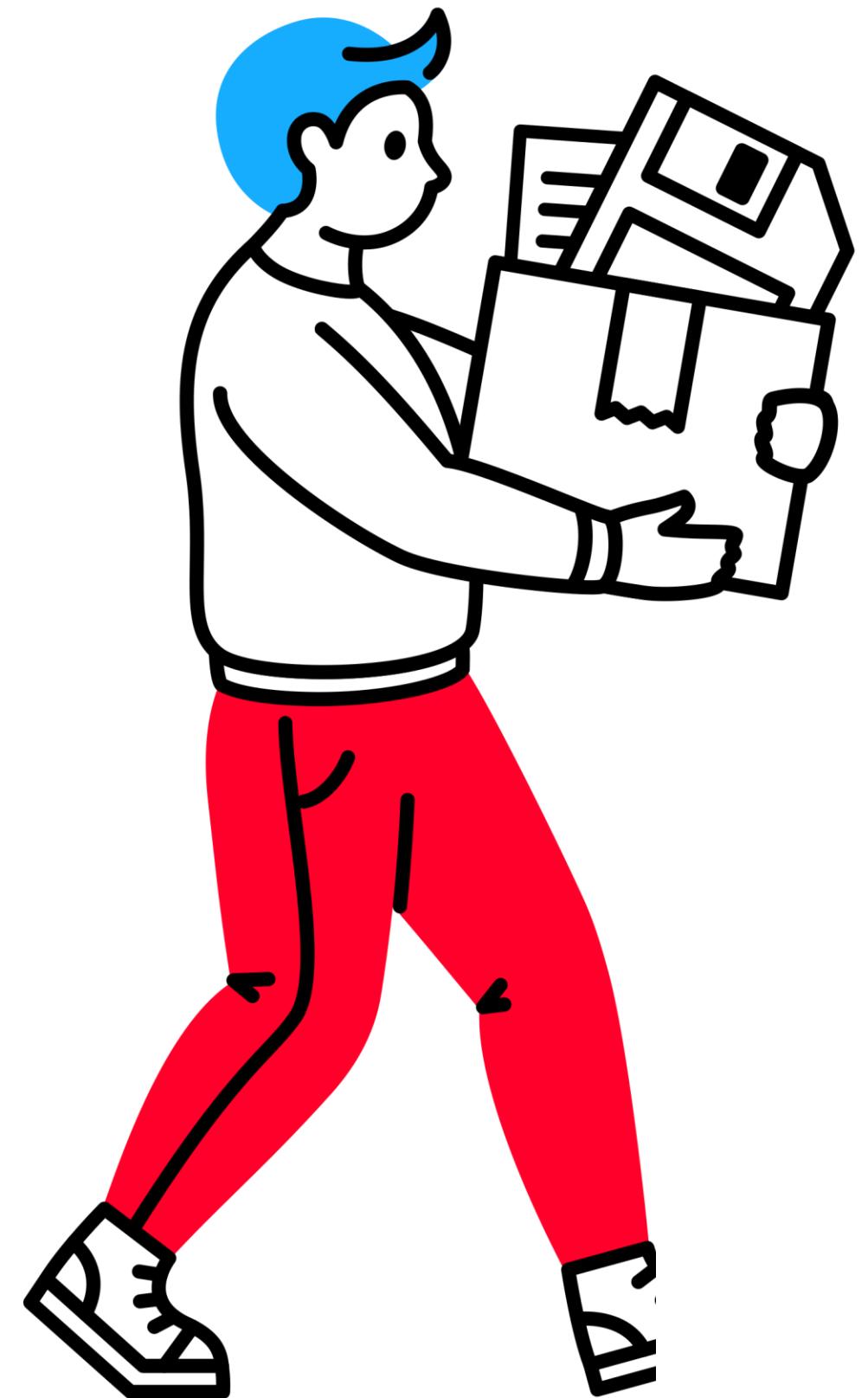
# Back in the Day...

- About 30 years ago

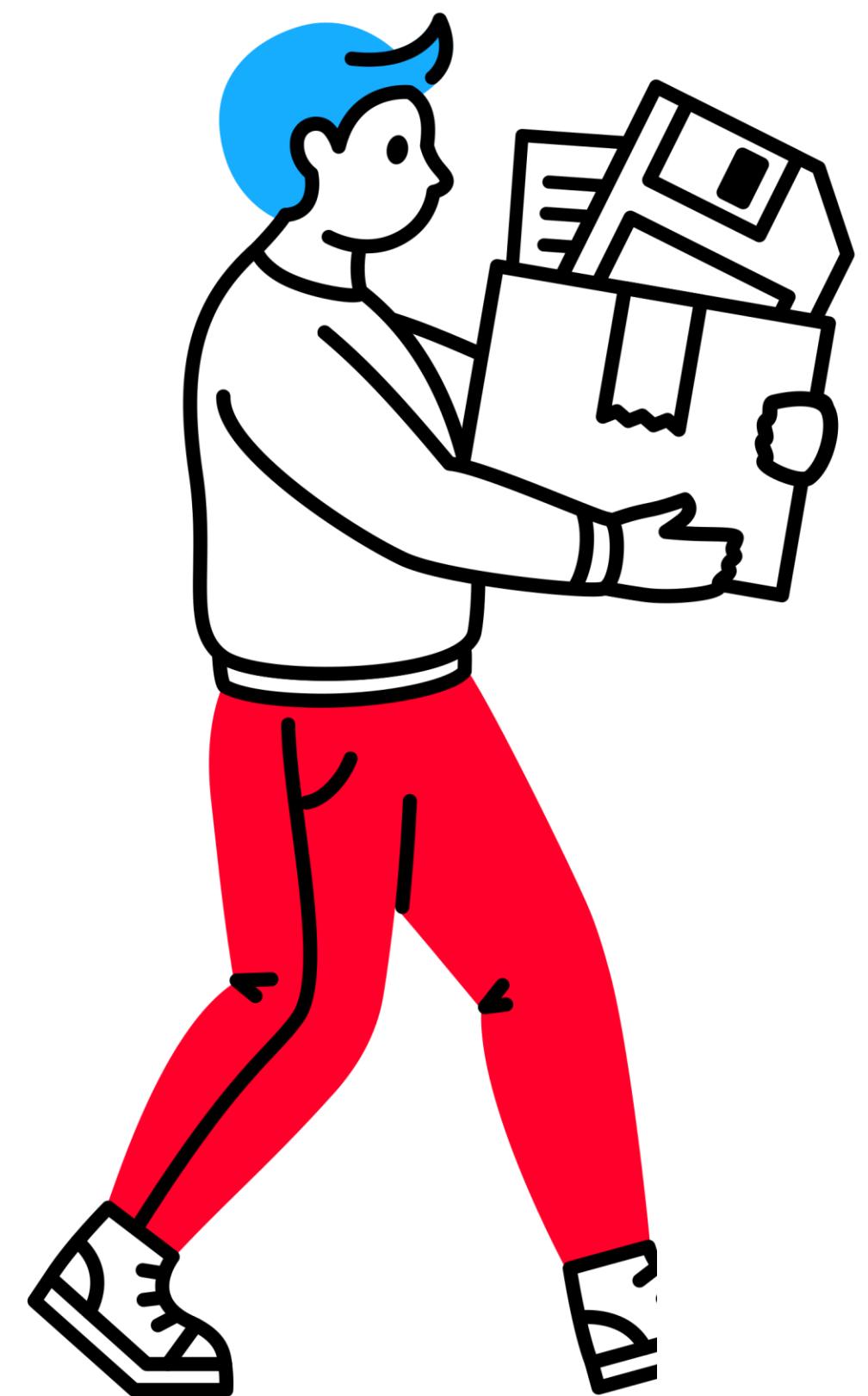


# Back in the Day...

- About 30 years ago
- When we needed to share files, and the Ethernet network was down

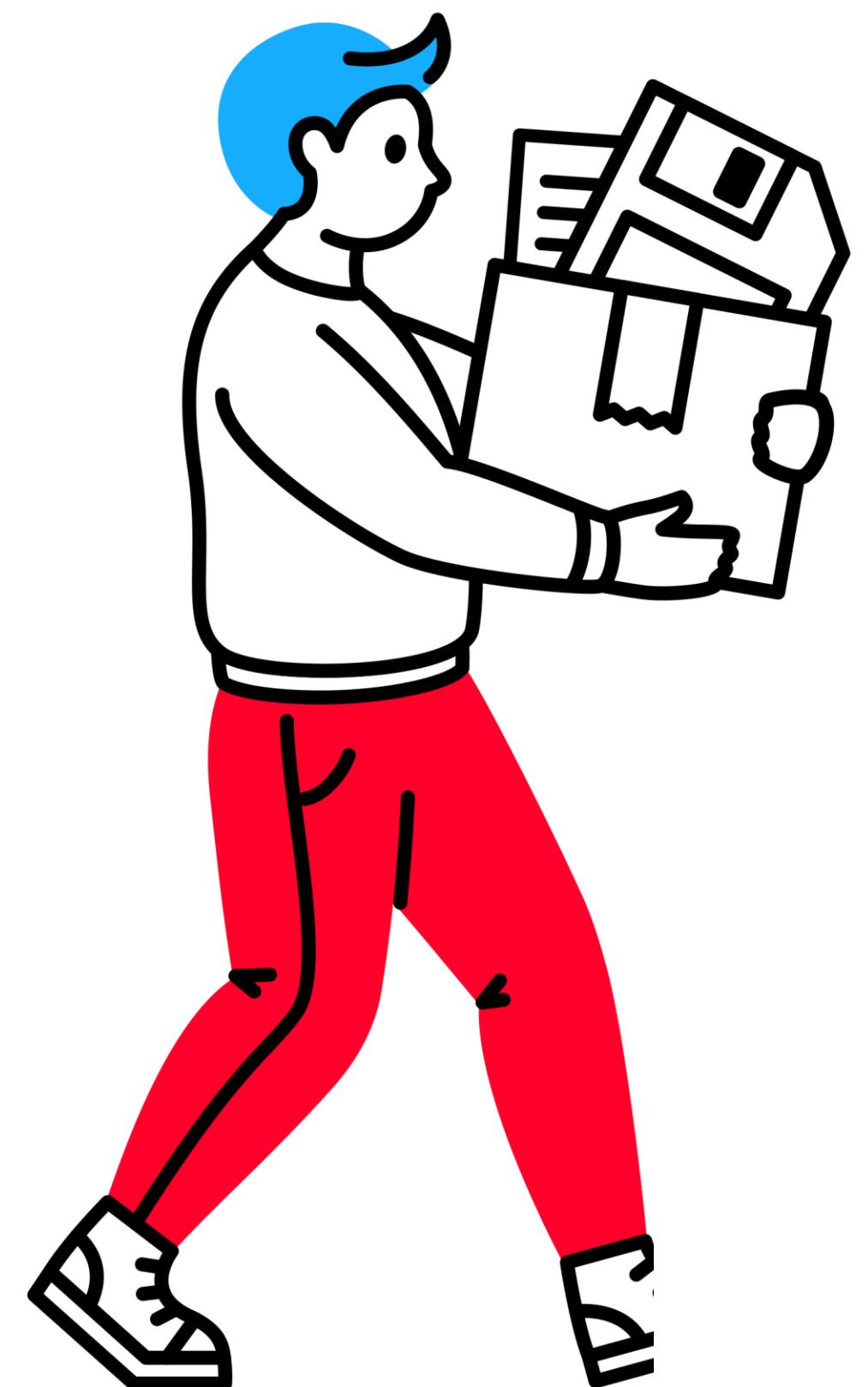


# Back in the Day...



- About 30 years ago
- When we needed to share files, and the Ethernet network was down
- We shared files via “sneaker net”

# Back in the Day...



- About 30 years ago
- When we needed to share files, and the Ethernet network was down
- We shared files via “sneaker net”
- So this was our “transport”

# Transports: STDIO

- STDIO

20

# Transports: STDIO

- STDIO
  - Both the MCP Client and MCP Server **must be on the same machine**

2.

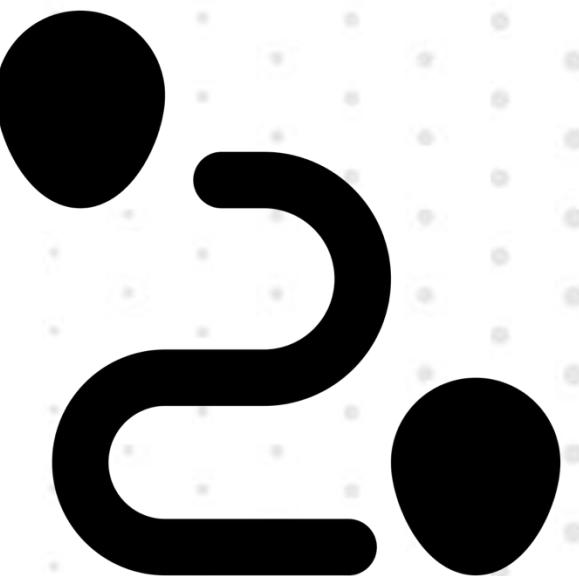
# Transports: STDIO

- STDIO
  - Both the MCP Client and MCP Server **must be on the same machine**
  - **Local** MCP Servers only

2.

# Transports: STDIO

- STDIO
  - Both the MCP Client and MCP Server **must be on the same machine**
  - **Local** MCP Servers only
  - This means that you can't use Standard I/O for logging



# Transports: Streamable HTTP

- Streamable HTTP

2.

# Transports: Streamable HTTP

- Streamable HTTP
  - The MCP Client and MCP Server can be on the different machines

2.

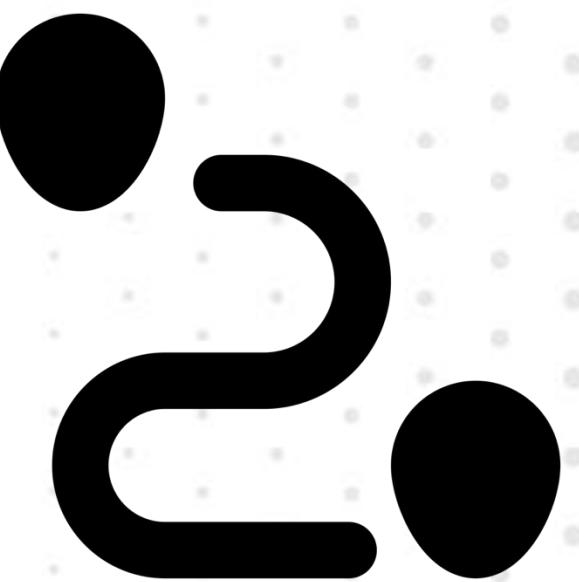
# Transports: Streamable HTTP

- Streamable HTTP
  - The MCP Client and MCP Server can be on the different machines
  - **Local and Remote** MCP Servers



# Transports: Streamable HTTP

- Streamable HTTP
  - The MCP Client and MCP Server can be on the different machines
  - **Local and Remote** MCP Servers
- Do you remember the MCP Client apps mentioned earlier?



# ChatGPT Playground

Prompts Playground - OpenAI API

BT Authors / O'Reilly Course AI Agents

Playground Dashboard Docs API reference

PLAYGROUND Prompts Images Realtime Assistants TTS

Prompts Your prompts Save

Add tools from remote MCP servers

Select from a list of popular server or connect to a new one

+ Add new Zapier Shopify

Intercom Stripe Plaid

Square Cloudflare Browser HubSpot

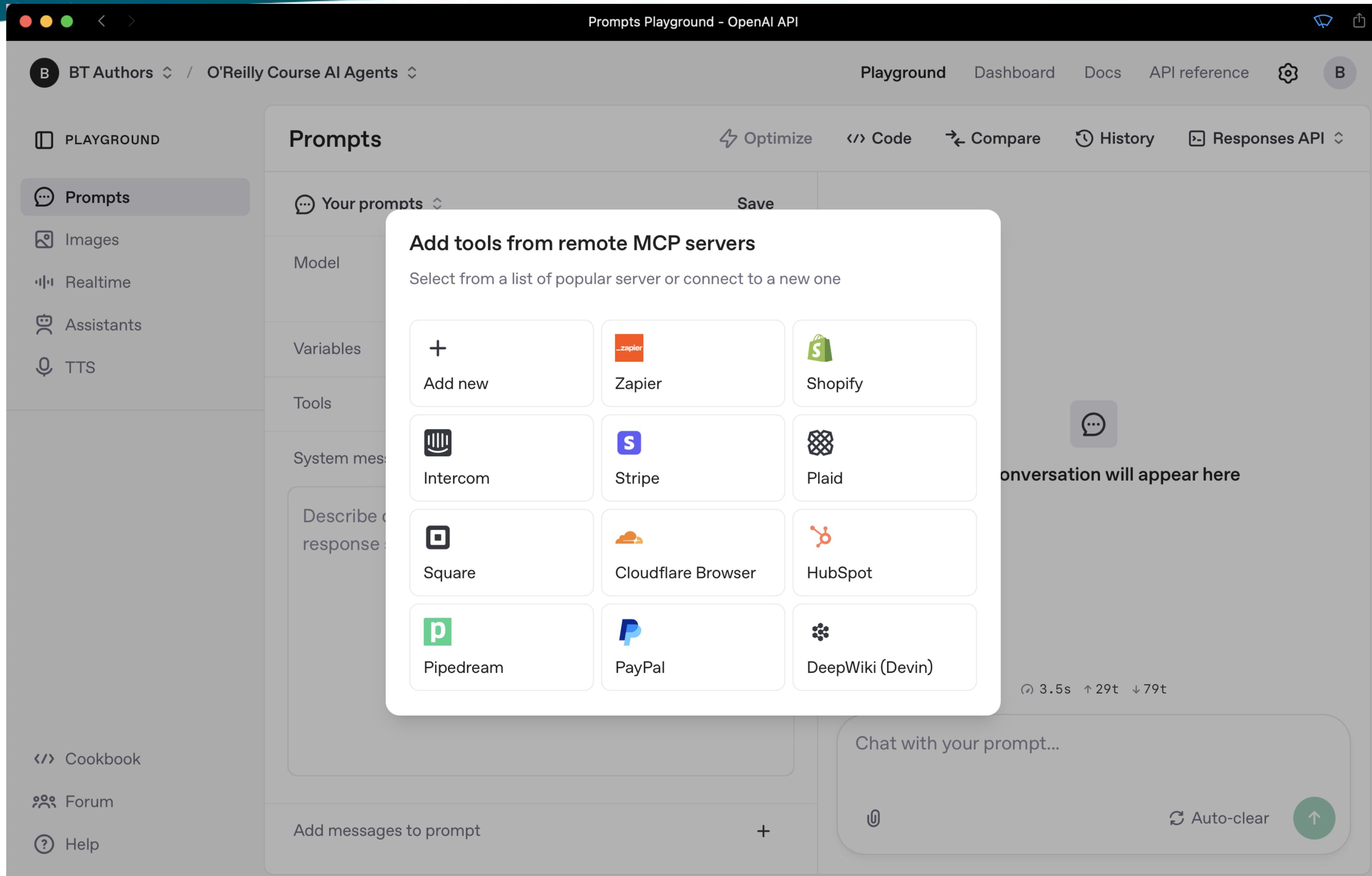
Pipedream PayPal DeepWiki (Devin)

onversation will appear here

3.5s ↑ 29t ↓ 79t

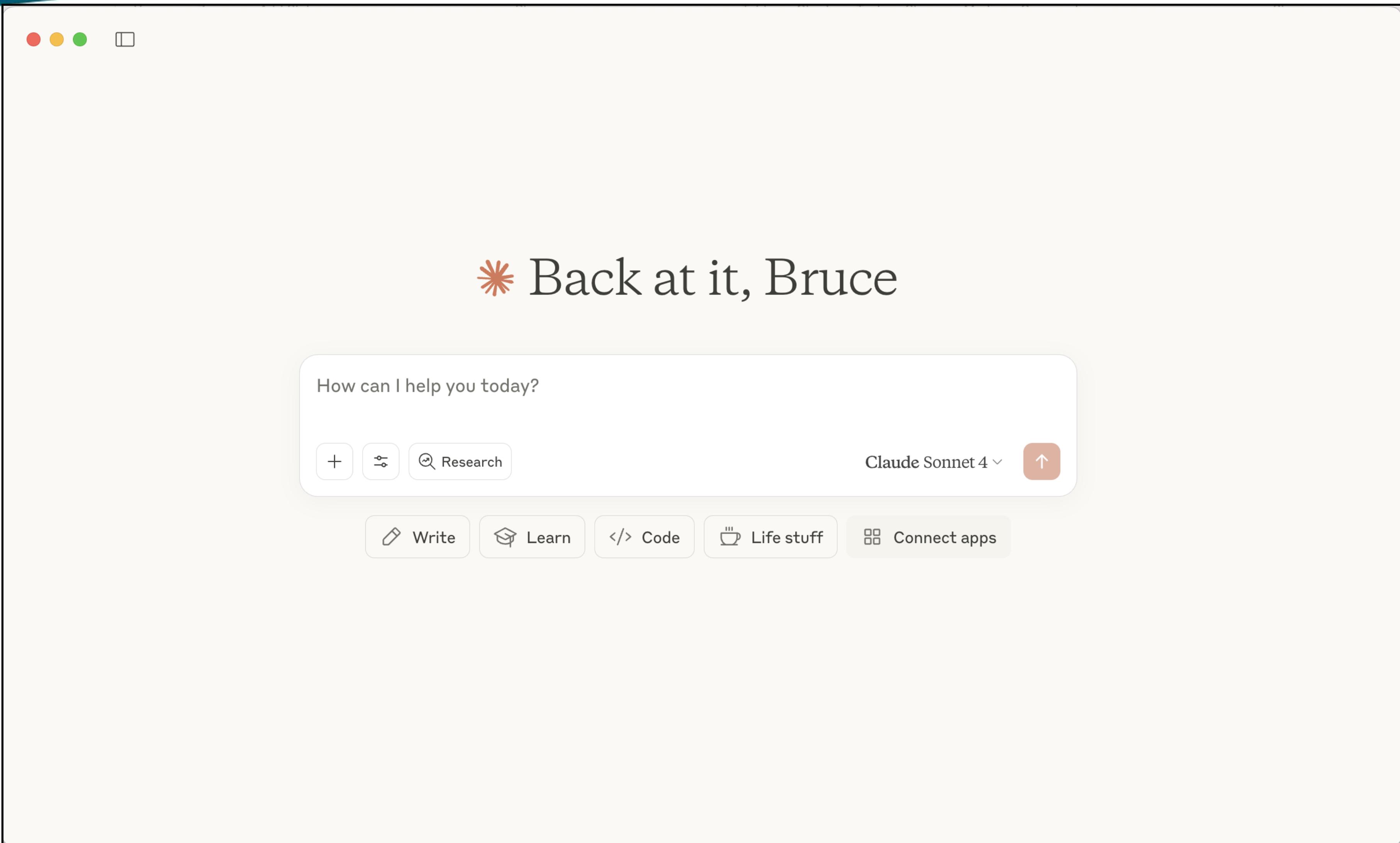
Chat with your prompt... Auto-clear

Add messages to prompt +



Remote

# Claude Desktop

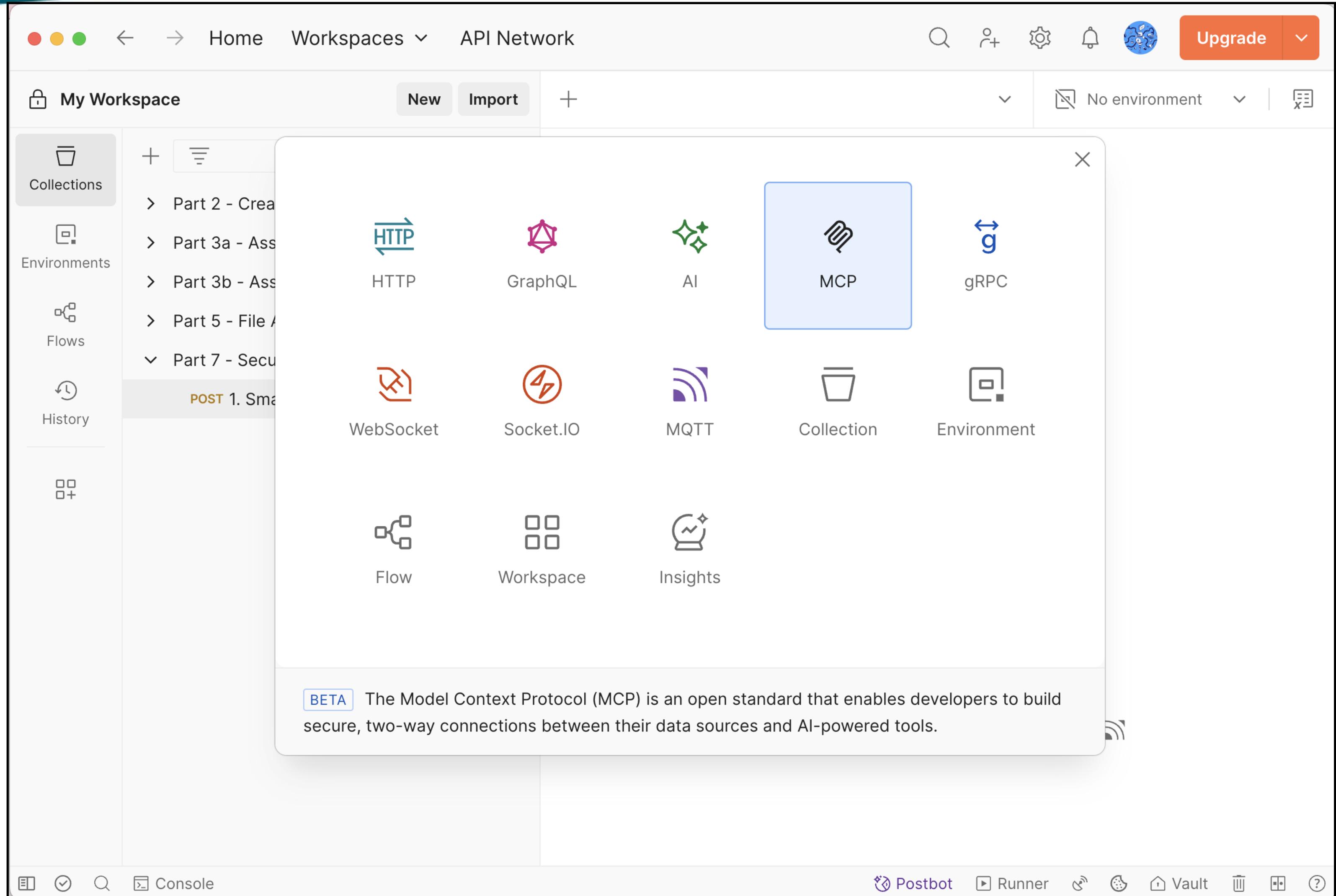


Local



Remote

# Postman



The screenshot shows the Postman application interface. On the left, there's a sidebar with 'My Workspace' containing 'Collections', 'Environments', 'Flows', and 'History'. The main area displays various API testing tools: 'HTTP', 'GraphQL', 'AI', 'MCP' (which is highlighted with a blue box), 'gRPC', 'WebSocket', 'Socket.IO', 'MQTT', 'Collection', 'Environment', 'Flow', 'Workspace', and 'Insights'. A tooltip for the MCP icon states: 'BETA The Model Context Protocol (MCP) is an open standard that enables developers to build secure, two-way connections between their data sources and AI-powered tools.' At the bottom, there are icons for 'Console', 'Postbot', 'Runner', 'Vault', and a question mark icon.



Local



Remote

# MCP Inspector

MCP Inspector v0.14.2

Transport Type

STDIO

Command

Command

Arguments

Arguments (space-separated)

> Environment Variables

Server Entry Servers File

> Configuration

Connect

Disconnected

History

No history yet

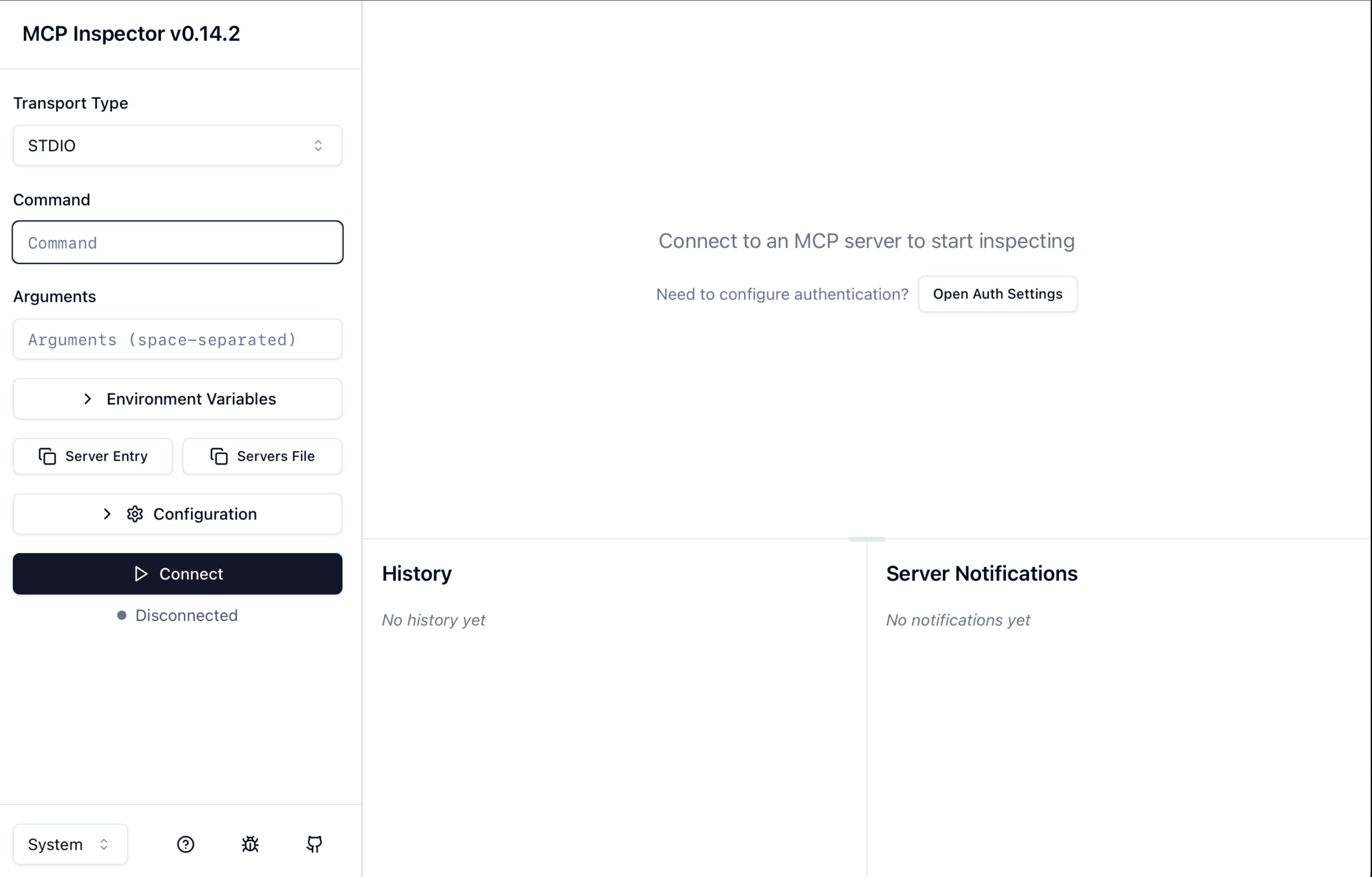
System ⓘ ⚡ ⌂

Server Notifications

No notifications yet

Connect to an MCP server to start inspecting

Need to configure authentication? [Open Auth Settings](#)



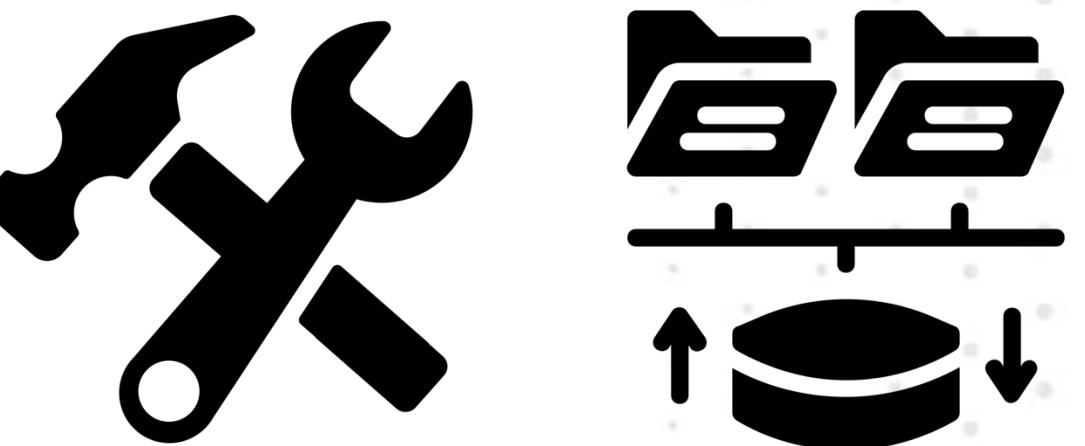
Local



Remote

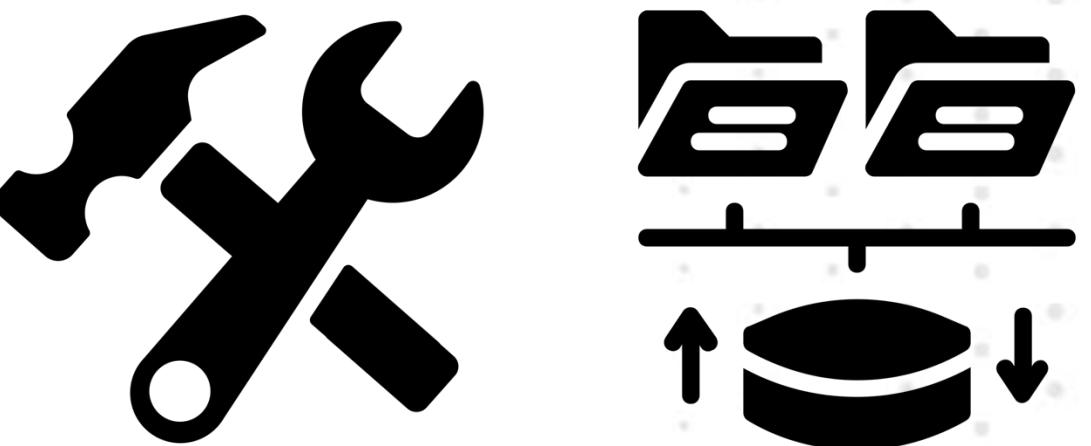
# Tools and Resources

- We need to cover these together



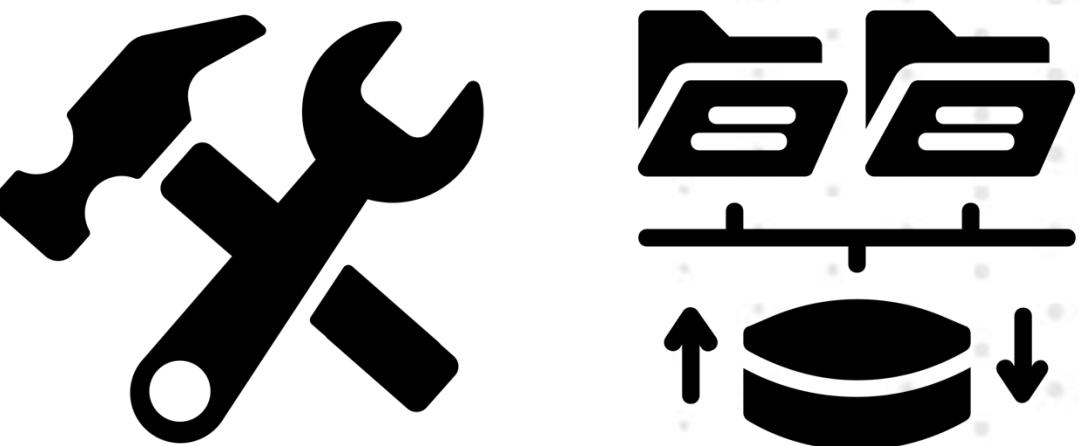
# Tools and Resources

- We need to cover these together
- They are closely related



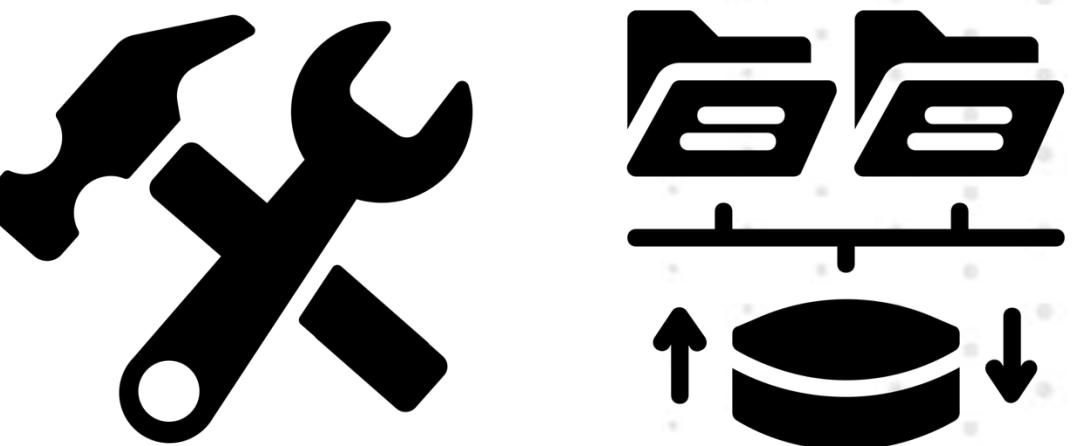
# Tools and Resources

- We need to cover these together
- They are closely related
- Do you remember the whole point of MCP?



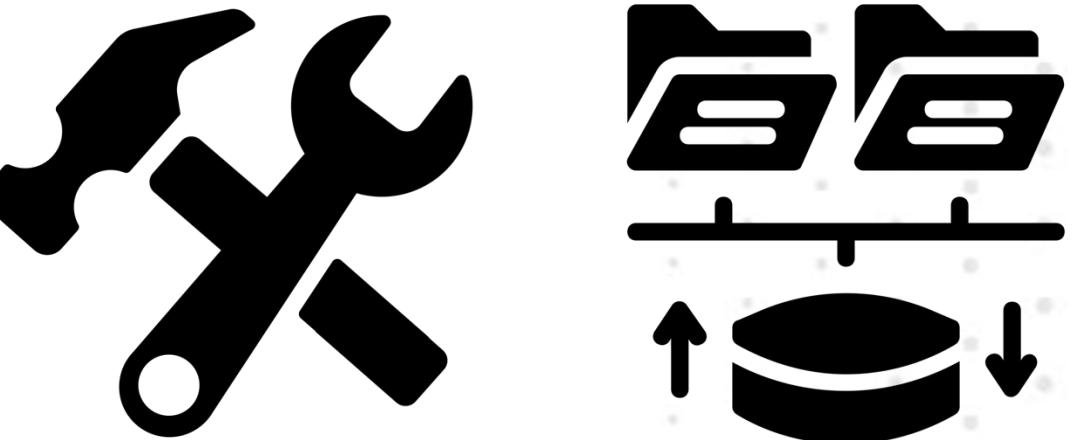
# The whole point of MCP

- (1) Use any compatible Client that you want



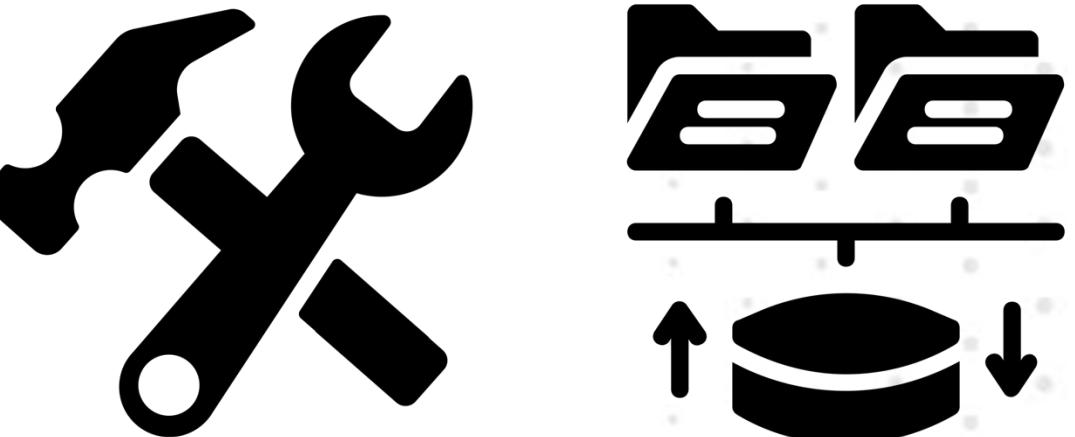
# The whole point of MCP

- (1) Use any compatible Client that you want
- (2) Use (or build) a Server to **AI-Enable the thing** that you want



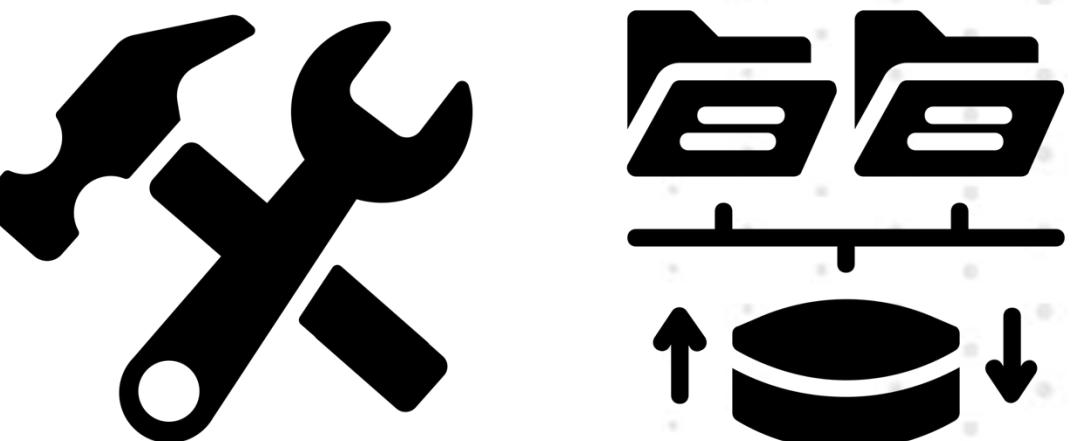
# The whole point of MCP

- (1) Use any compatible Client that you want
- (2) Use (or build) a Server to **AI-Enable the thing** that you want
- We saw this with the **Weather Demo**

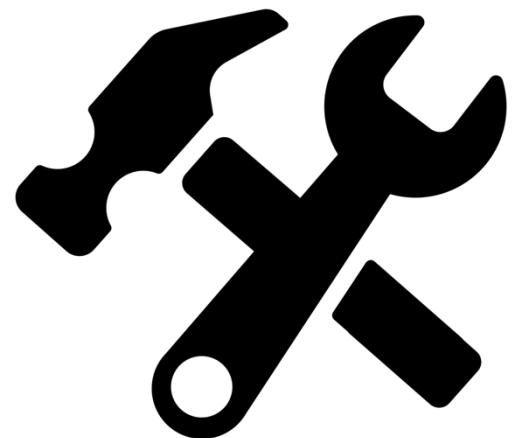


# The whole point of MCP

- (1) Use any compatible Client that you want
- (2) Use (or build) a Server to **AI-Enable the thing** that you want
- We saw this with the **Weather Demo**
- The AI-Enabled thing is either a **Tool** or a **Resource**



# When is it a Tool?



- The “thing” needs to read a value - like the current weather

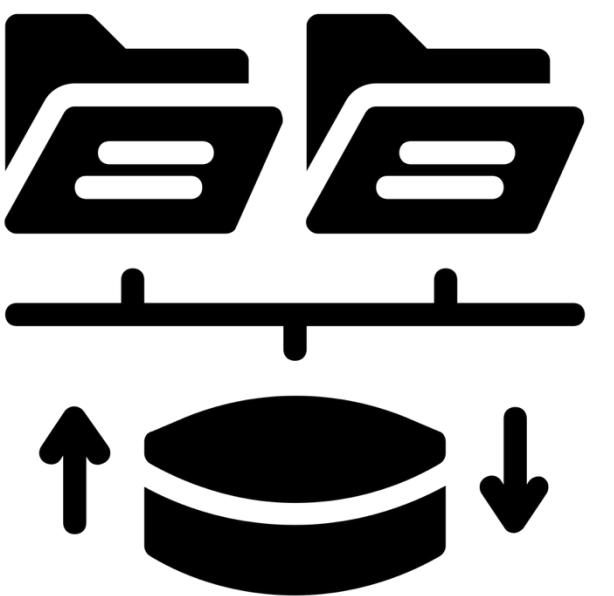
# When is it a Tool?



- The “thing” needs to read a value - like the current weather
- The “thing” needs to do some action - like, turn on a light, or start/stop playing a song

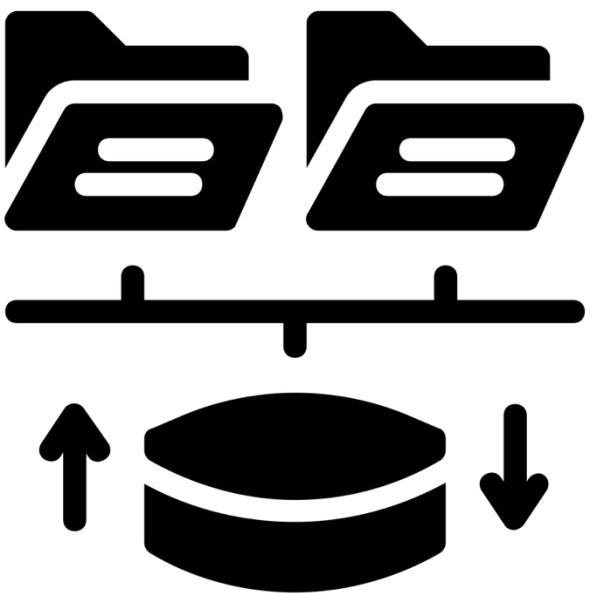
# When is it a Resource?

- The “thing” needs to access a repository of data



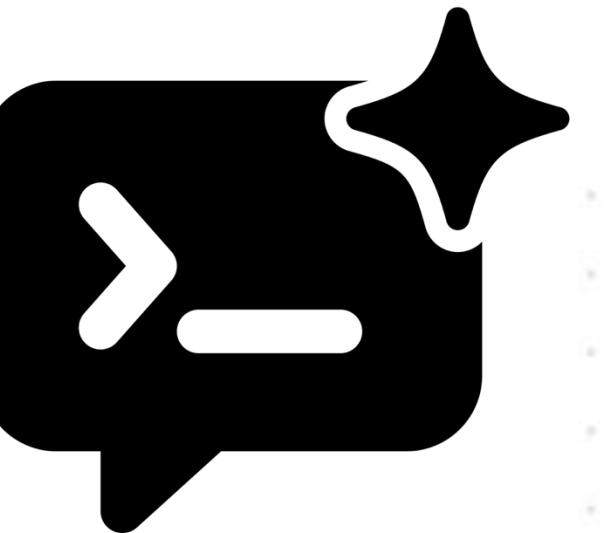
# When is it a Resource?

- The “thing” needs to access a repository of data
- Think of things like:
  - file systems
  - databases
  - code repos



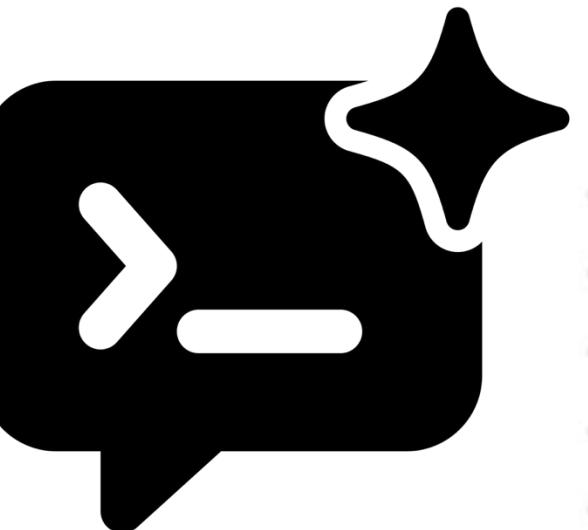
# Prompts

- How does the AI model with the MCP Client know what your MCP server can do?



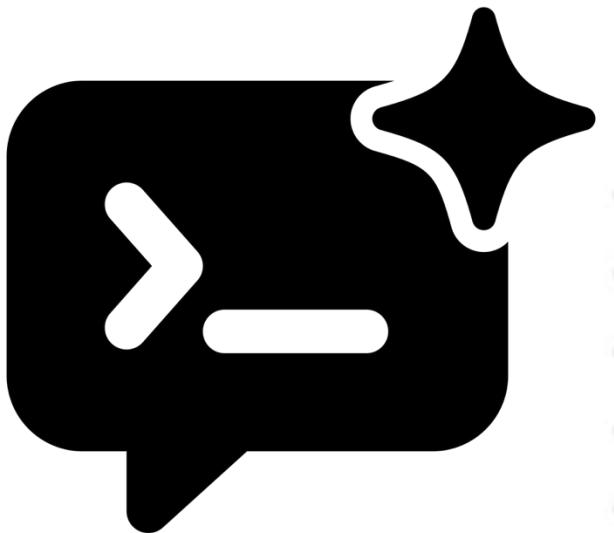
# Prompts

- How does the AI model with the MCP Client know what your MCP server can do?
- Prompts



# Prompts

- How does the AI model with the MCP Client know what your MCP server can do?
- Prompts
- Prompt Specifications, which are example prompts for AI model to read



# Questions?





## PART 3

### Creating an MCP Server in Python



# What have we accomplished so far?

- We are familiar with the terms and terminology for MCP
- We know how MCP clients work
- We know how MCP servers work



# What can be improved upon?



- We need to see how to build an MCP Server in Python
- We need to see how to test an MCP Server with the MCP Inspector

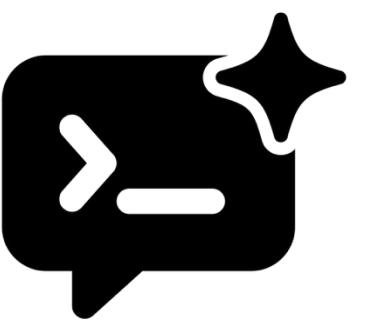
## Official MCP SDKs



# Remember This Slide?



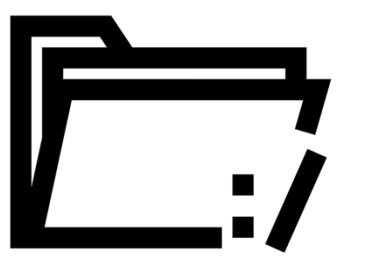
- Message Protocol



- Prompts



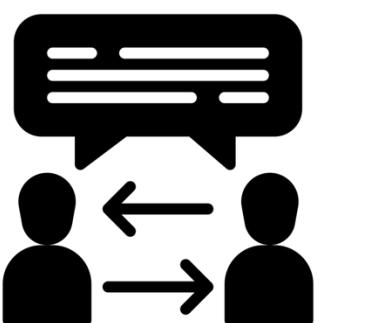
- Transports



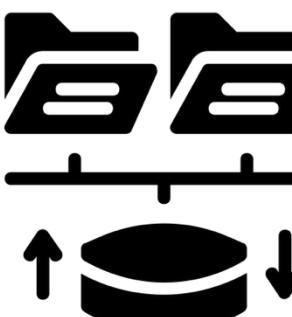
- Roots



- Tools



- Sampling



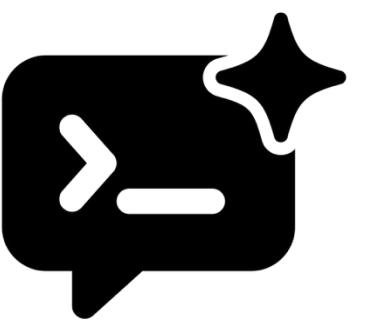
- Resources



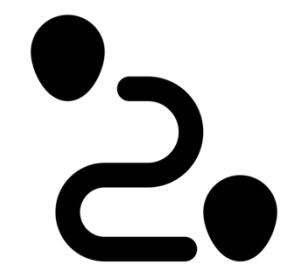
# Remember This Slide?



- Message Protocol



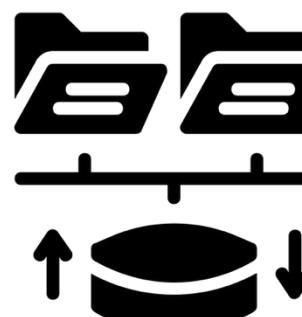
- Prompts



- Transports



- Tools



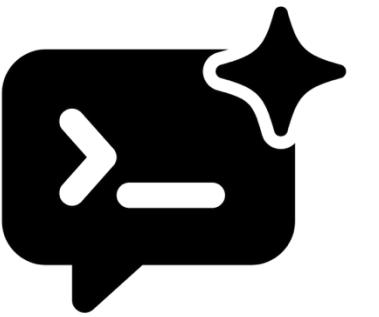
- Resources

- ~~Roots~~
- ~~Sampling~~



# Implementation Details

- ~~Message Protocol~~



- Prompts

- Transports

- Tools

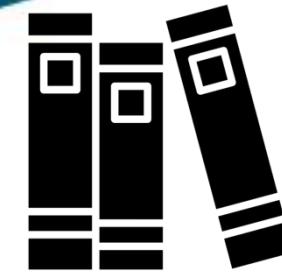
- Resources

- ~~Roots~~

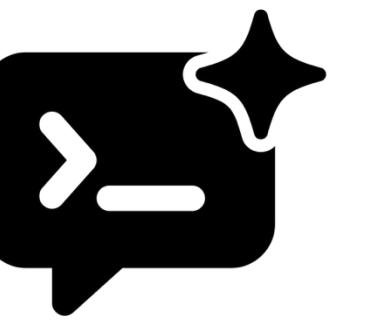
- ~~Sampling~~



# Implementation Details



- Dependencies



- Prompts



- Transports



- Tools

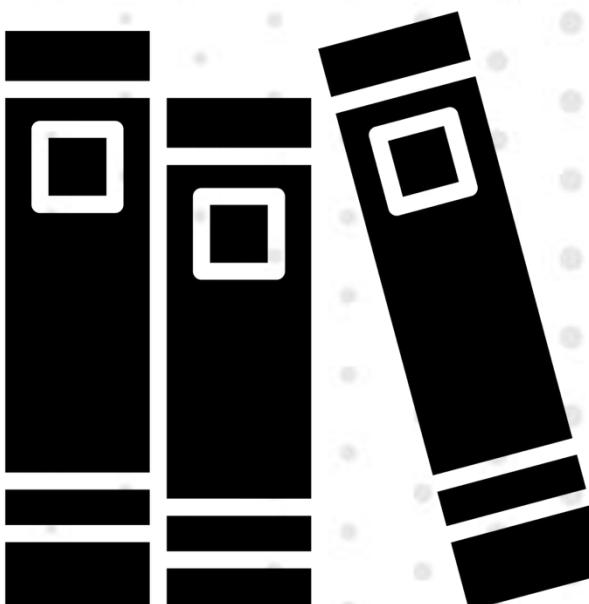


# Dependencies

```
from mcp.server.fastmcp import FastMCP  
from mcp.server.fastmcp.prompts import base
```

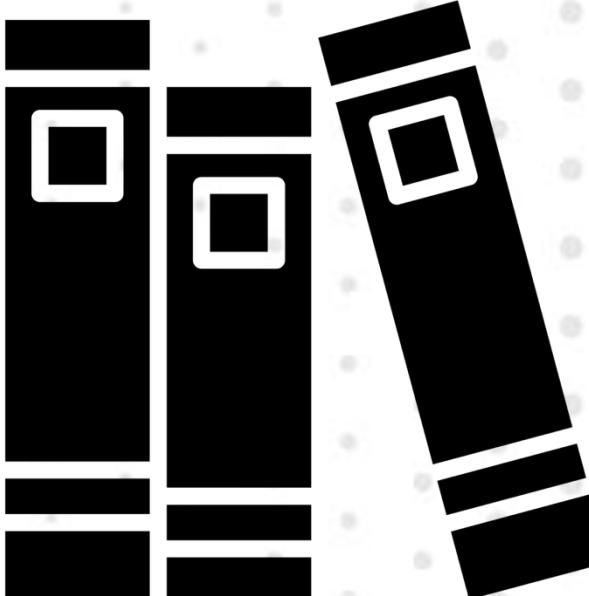
- (part of official SDK)

```
#!/usr/bin/env python3  
....  
  
Weather MCP Server - A Model Context Protocol server that provides weather information.  
This server implements:  
- A weather tool that returns hardcoded temperature data (45F)  
- MCP prompt templates for weather-related interactions  
- STDIO transport for communication with Claude Desktop  
  
Built using FastMCP (included in the official MCP Python SDK).  
....  
  
import sys  
from datetime import datetime  
  
from mcp.server.fastmcp import FastMCP  
from mcp.server.fastmcp.prompts import base
```



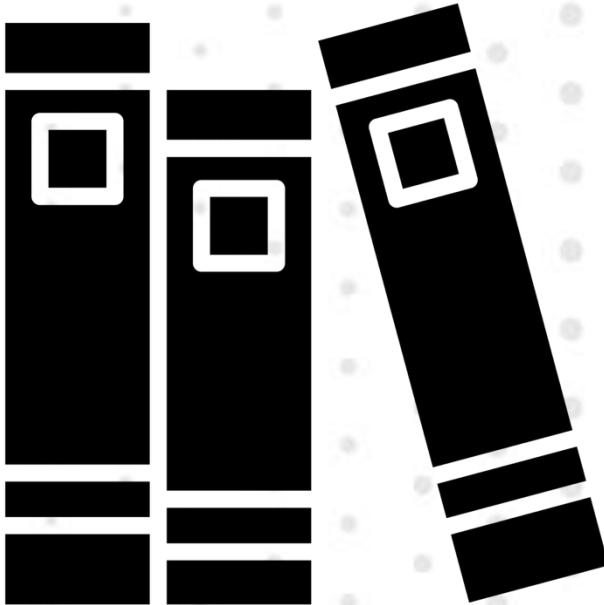
# uv vs pip: Which is better?

- I'm using UV as my package manager
- UV has several advantages:
  - (1) built-in environment management.  
No need to create a virtual environment
  - (2) up to 100x faster than pip for  
package management and dependency  
resolution



# Installation

- `uv add "mcp[cli]"`
- You want the CLI extras, which allows you to have more commands at the command line



# Create the MCP Server



# Create the MCP Server

```
mcp = FastMCP("weather-server")
```



# Create the MCP Server

```
mcp = FastMCP("weather-server")
```

```
# Create the FastMCP server
mcp = FastMCP("weather-server")
```



# Setting the Transport

2.

# Setting the Transport

`mcp.run()`

2.

# Setting the Transport

## mcp.run()

```
def main():
    """
    Main entry point for the Weather MCP Server.
    Sets up STDI0 transport and starts the server to listen for MCP requests.
    """

    try:
        # Start the server with STDI0 transport
        print("Starting Weather MCP Server...", file=sys.stderr)
        print("Server ready to accept connections via STDI0", file=sys.stderr)

        # Run the server - FastMCP handles all the transport details
        mcp.run()

    except Exception as e:
        # Log error to stderr (stdout is used for MCP communication)
        print(f"Failed to start Weather MCP Server: {e}", file=sys.stderr)
        import traceback
        traceback.print_exc(file=sys.stderr)
        sys.exit(1)
```



# Declaring our Tool for Weather



# Declaring our Tool for Weather

@mcp.tool()



# Declaring our Tool for Weather

@mcp.tool()

```
@mcp.tool()
def get_weather(city: str) -> str:
    """
    Get current weather information for a specified city.
    Returns temperature data for the requested location.

    Args:
        city: Name of the city to get weather for (e.g., 'New York', 'London', 'Tokyo')

    Returns:
        Formatted weather information string
    """
    # Validate city parameter
    if not city or not city.strip():
        raise ValueError("City parameter cannot be empty")

    city = city.strip()
```



# Declaring our Tool for Weather

@mcp.tool()

```
city = city.strip()

# Log the request for debugging (to stderr to avoid interfering with STDO)
print(f"Processing weather request for city: {city}", file=sys.stderr)

# Get current timestamp for the response
timestamp = datetime.now().strftime("%Y-%m-%d %H:%M:%S")

# Create formatted weather response
# NOTE: This is hardcoded for now – will be replaced with actual API calls
weather_report = (
    f"Weather Report for {city}\n"
    "=====\\n"
    f"Current Temperature: 83F\\n"
    "Conditions: Clear\\n"
    "Humidity: 65%\\n"
    "Wind: Light breeze\\n"
    f"Last Updated: {timestamp}\\n"
    "\\n"
)

# Log the response for debugging
print(f"Returning weather data for {city}", file=sys.stderr)

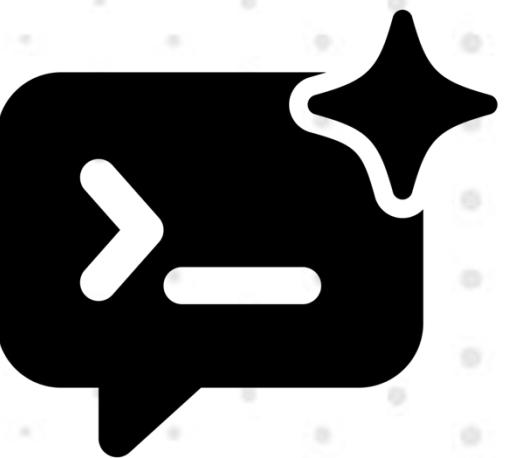
return weather_report
```



Pearson



# Declaring our Prompts



# Declaring our Prompts

@mcp.prompt()



# Declaring our Prompts

@mcp.prompt()

```
@mcp.prompt()
def weather_inquiry(location: str) -> str:
    """
    Template for asking about weather conditions in a specific location.

    Args:
        location: The city or location to inquire about

    Returns:
        Formatted prompt for weather inquiries
    """

    return (
        f"I need current weather information for {location}. "
        "Please provide the temperature and any relevant weather conditions. "
        "If you need to use a tool to get this information, please do so."
    )
```



# Declaring our Prompts

## @mcp.prompt()

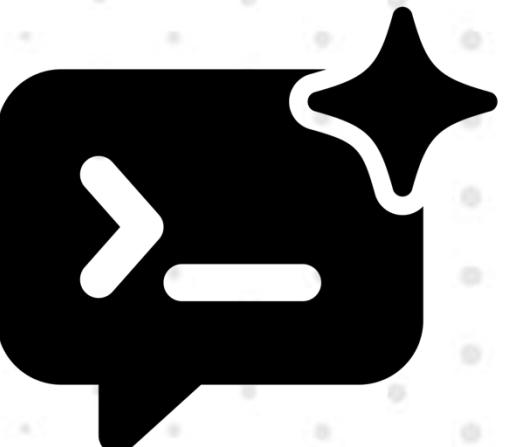
```
@mcp.prompt()
def weather_travel_advice(destination: str, travel_date: str = None) -> list[base.Message]:
    """
    Template for getting weather-based travel advice for a destination.

    Args:
        destination: Travel destination city
        travel_date: Planned travel date (optional)

    Returns:
        List of messages for travel weather advice
    """
    date_info = f" for travel on {travel_date}" if travel_date else " for current conditions"

    prompt_text = (
        f"I'm planning to travel to {destination}{date_info}. "
        "Please check the current weather conditions and provide advice on "
        "what to pack and any weather-related considerations for my trip. "
        "Use the weather tool to get current temperature data."
    )

    return [
        base.UserMessage(prompt_text)
    ]
```



## DEMO 3

# Weather MCP Server - Python

## DEMO 3

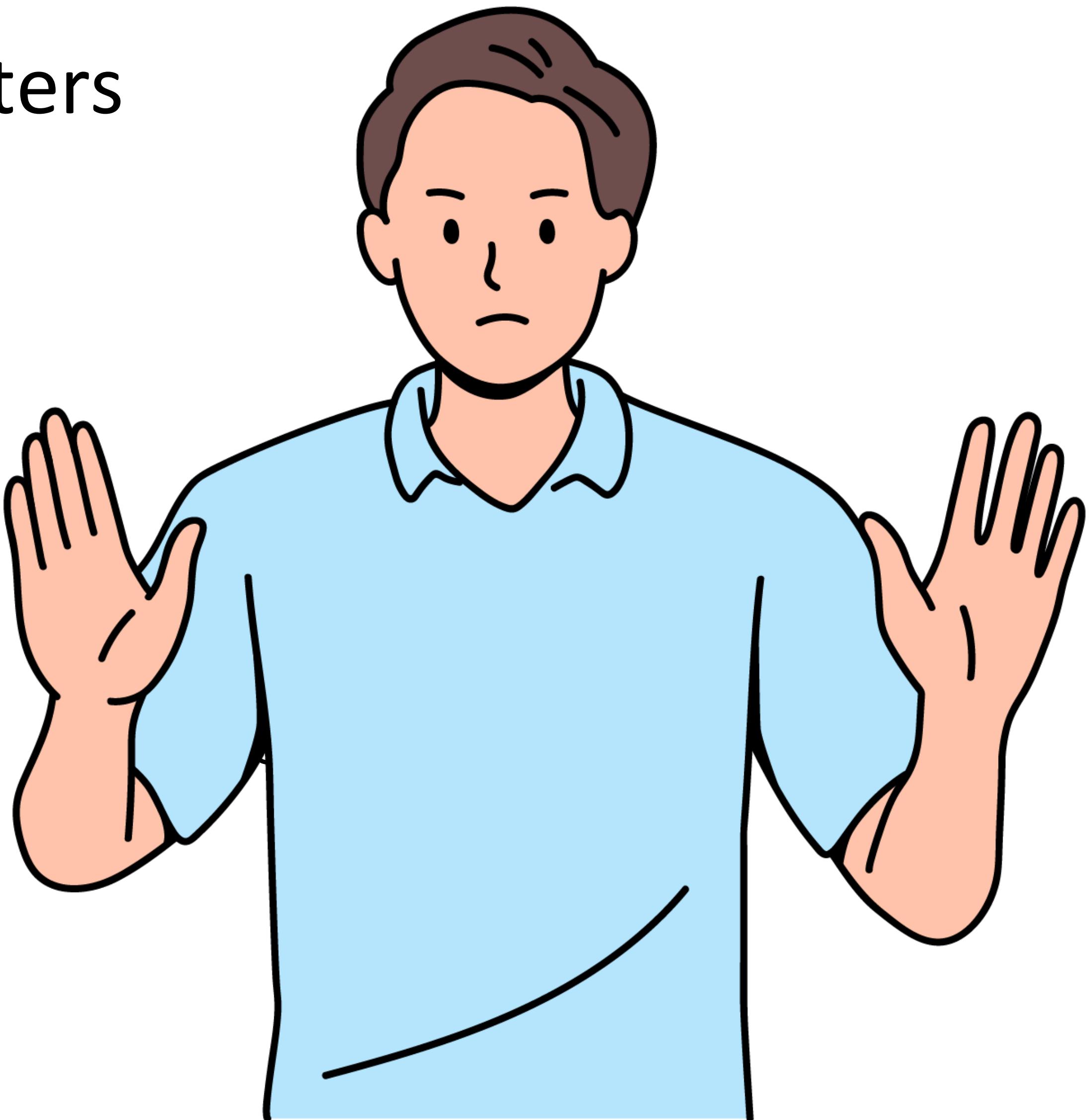


MCP Inspector



# Let's Talk About Transports Again

- The Transport Matters



2.

# Let's Talk About Transports Again

- The Transport Matters

Client



MCP Inspector

Server



Weather MCP

# Let's Talk About Transports Again

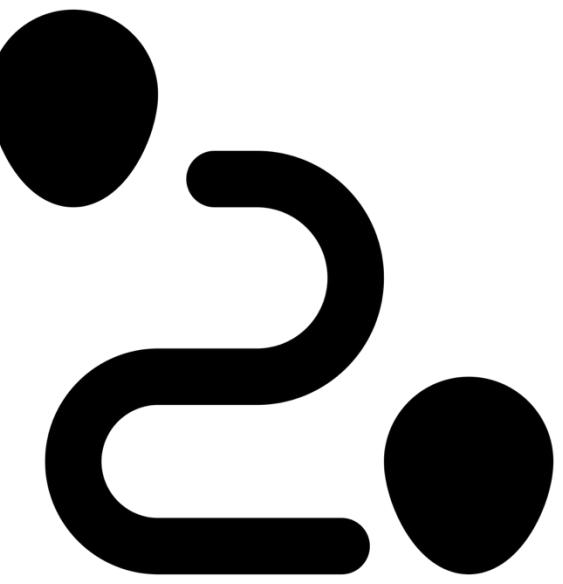
- The Transport Matters

Client



MCP Inspector

Transport



HTTP

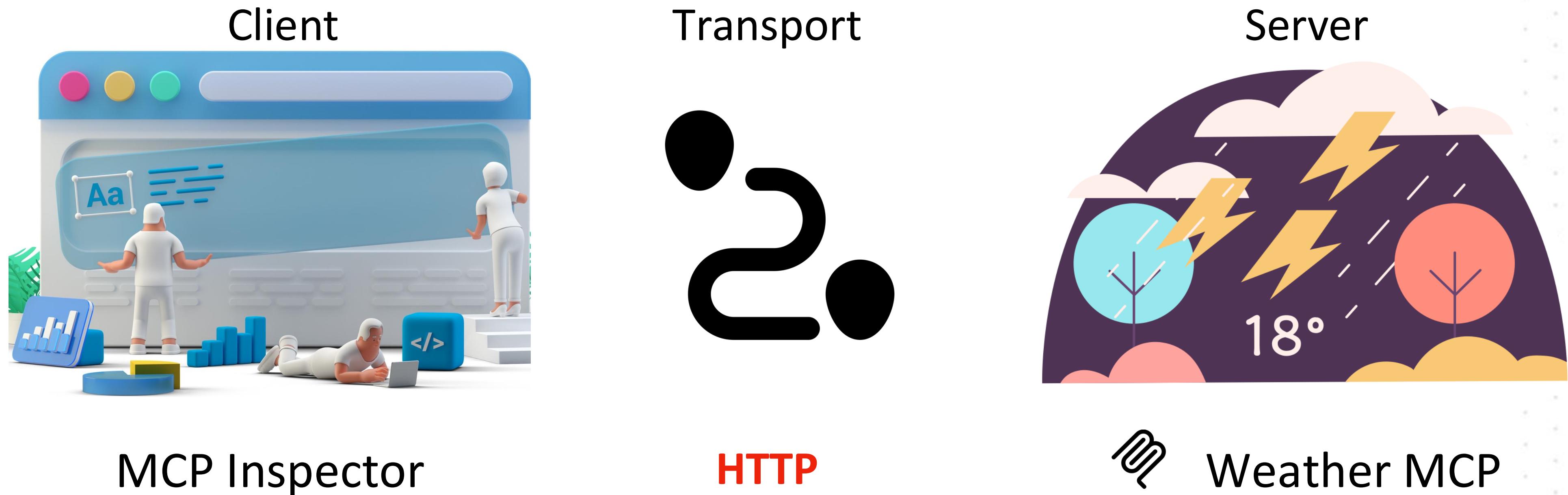
Server



 Weather MCP

# Let's Talk About Transports Again

- The Transport Matters



The Server needs to be running **first**

# Let's Talk About Transports Again

- The Transport Matters

Client



MCP Inspector

Server



Weather MCP

# Let's Talk About Transports Again

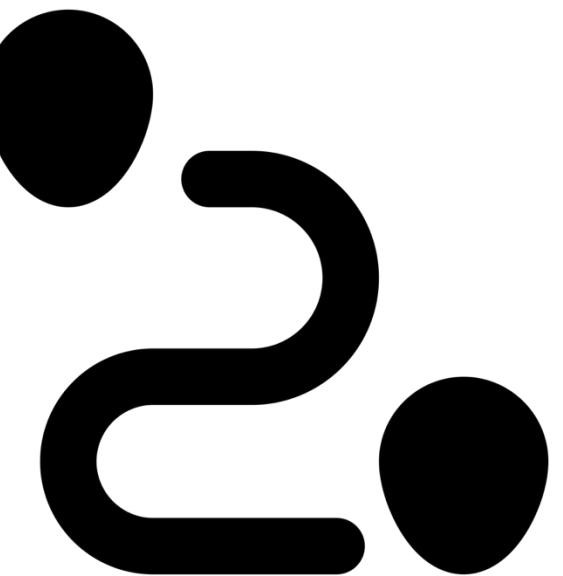
- The Transport Matters

Client



MCP Inspector

Transport



STUDIO

Server



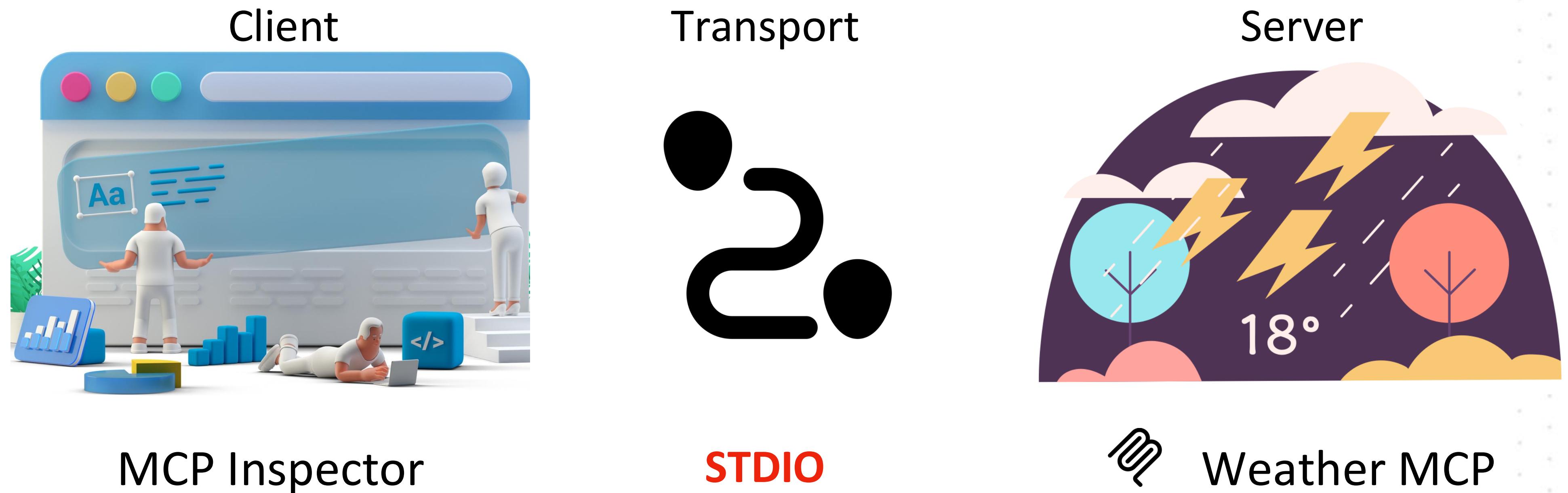
Weather MCP



Pearson

# Let's Talk About Transports Again

- The Transport Matters



No need to start the Server. The **client starts the server**

# Weather MCP Server - Python

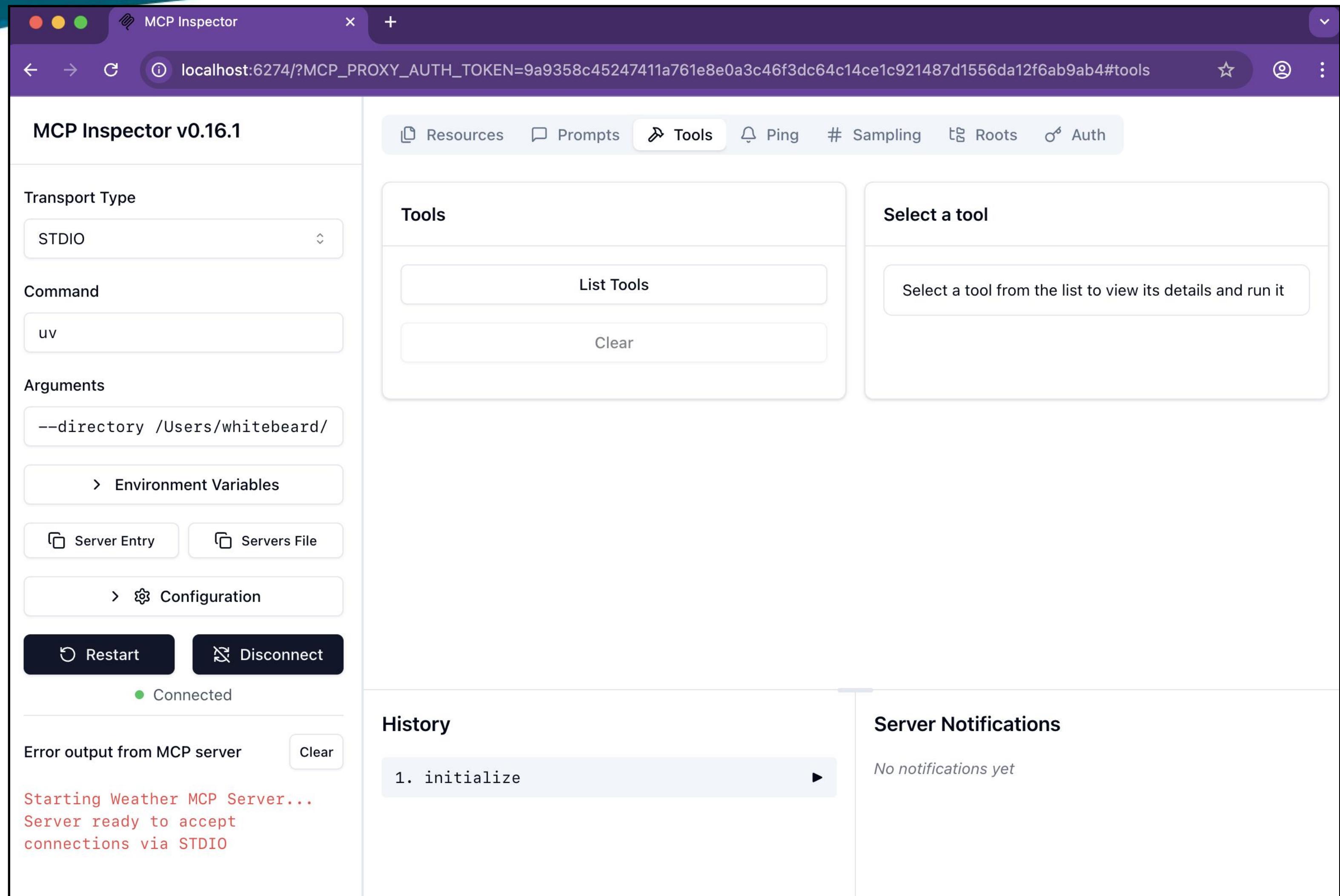
## DEMO 3



MCP Inspector



# MCP Inspector - How to Add a New MCP Server



MCP Inspector v0.16.1

Transport Type: STDIO

Command: uv

Arguments: --directory /Users/whitebeard/

> Environment Variables

Server Entry Servers File

> Configuration

Restart Disconnect

Connected

Error output from MCP server

Starting Weather MCP Server...  
Server ready to accept  
connections via STDIO

Resources Prompts Tools Ping Sampling Roots Auth

Tools

List Tools Clear

Select a tool

Select a tool from the list to view its details and run it

History

1. initialize

Server Notifications

No notifications yet



## MCP Inspector

# Weather MCP Server - Python

## DEMO 3



### PROMPT:

*What's the weather today in Seattle?*



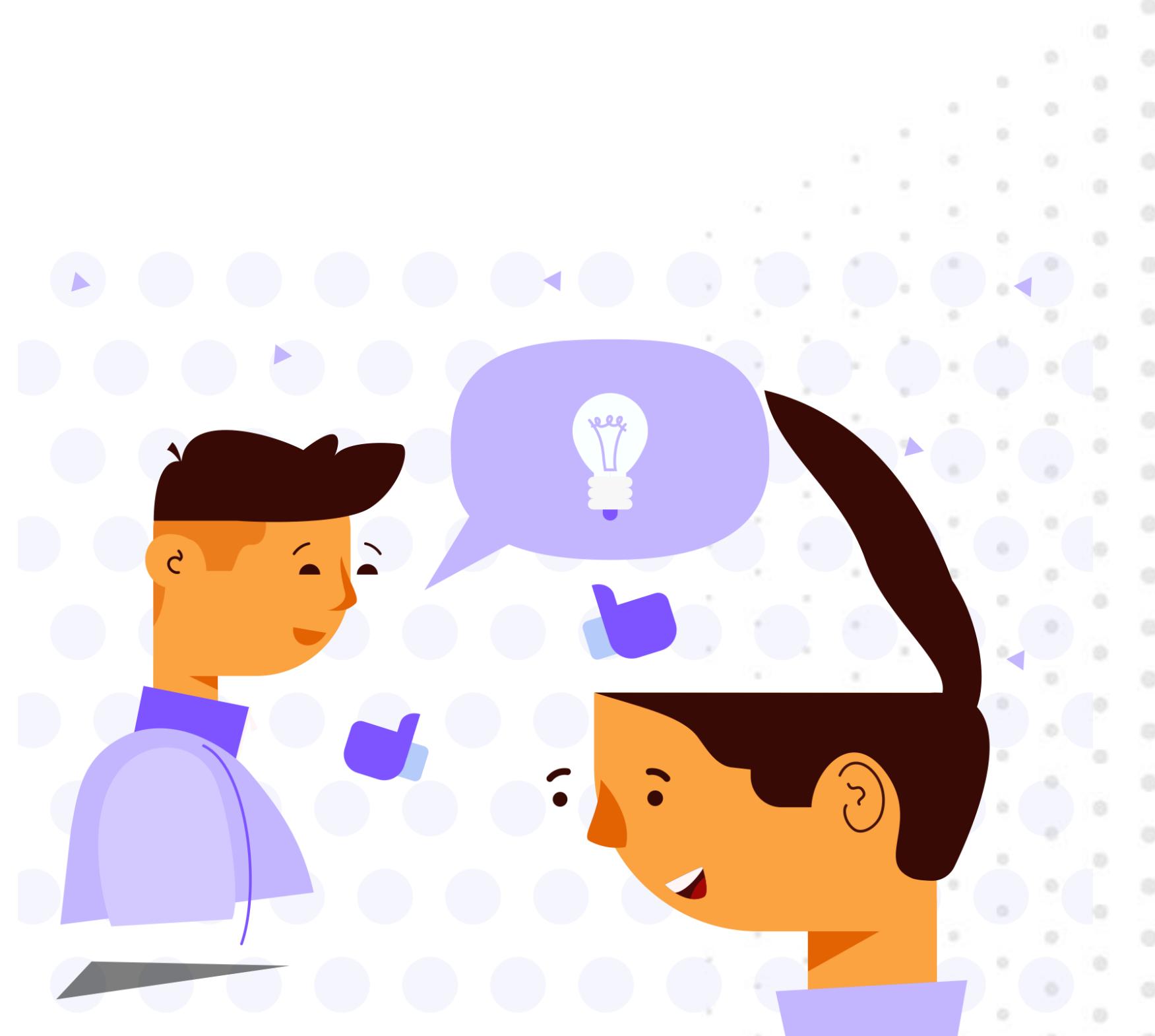
MCP Inspector



Weather MCP



# Questions?



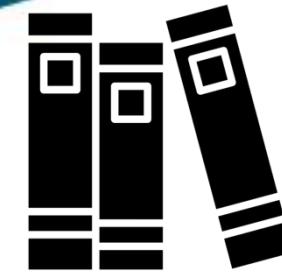


## PART 4

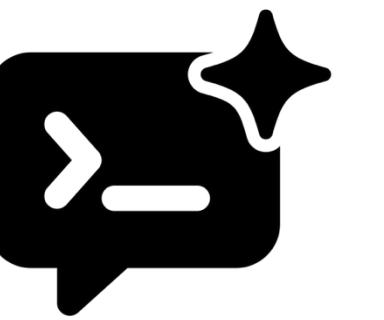
### Creating an MCP Server in Java



# Implementation Details



- Dependencies



- Prompts



- Transports

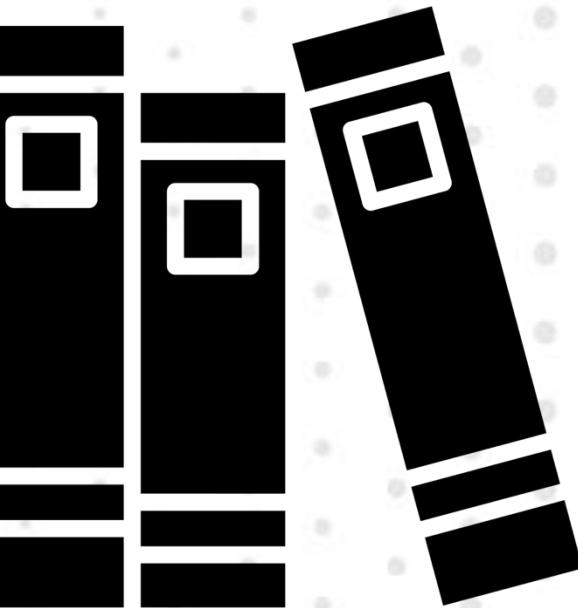


- Tools



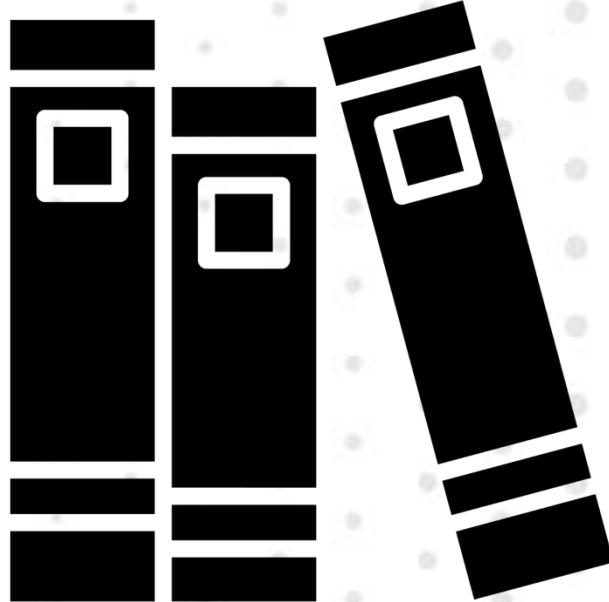
# Dependencies

```
<!-- MCP Java SDK -->  
<dependency>  
  <groupId>io.modelcontextprotocol.sdk</groupId>  
  <artifactId>mcp</artifactId>  
  <version>0.10.0</version>  
</dependency>
```



# Dependencies

```
import io.modelcontextprotocol.server.McpServer;
import io.modelcontextprotocol.server.McpServerFeatures;
import io.modelcontextprotocol.server.McpSyncServer;
import io.modelcontextprotocol.server.McpServerFeatures.SyncToolSpecification;
import io.modelcontextprotocol.server.transport.StdioServerTransportProvider;
import io.modelcontextprotocol.spec.McpSchema;
import com.fasterxml.jackson.databind.ObjectMapper;
```



# Create the MCP Server

```
/**  
 * Weather MCP Server - A Model Context Protocol server that provides weather information  
 * This server implements:  
 * - A weather tool that returns hardcoded temperature data (45F)  
 * - MCP prompt templates for weather-related interactions  
 * - STUDIO transport for communication with Claude Desktop  
 *  
 * Built using the official MCP Java SDK.  
 */  
public class WeatherMcpServer {
```



# Setting the Transport

# Setting the Transport

```
public static void main(String[] args) {
    try {
        // Create STUDIO transport provider for communication with Claude Desktop
        StdioServerTransportProvider transportProvider = new StdioServerTransportProvider(new ObjectMapper());

        // Create the weather tool specification
        SyncToolSpecification weatherToolSpec = createWeatherToolSpecification();

        // Create the MCP server with tools and prompts
        McpSyncServer server = McpServer.sync(transportProvider)
            .serverInfo(name:"weather-server", version:"1.0.0")
            .capabilities(McpSchema.ServerCapabilities.builder()
                .tools(listChanged:true)      // Enable tools support
                .prompts(listChanged:true)    // Enable prompts support
                .build())
            .tools(weatherToolSpec)
            .prompts(createWeatherPromptSpecifications())
            .build();
    }
}
```



# Declaring our Tool for Weather



# Declaring our Tool for Weather

```
/**  
 * Creates the weather tool specification that Claude can call to get weather information.  
 * The tool accepts a city parameter and returns hardcoded temperature data.  
 *  
 * @return Tool specification with handler  
 */  
private static McpServerFeatures.SyncToolSpecification createWeatherToolSpecification() {  
  
    // Define the schema as a JSON string  
    String schemaString = """"  
        {  
            "type": "object",  
            "properties": {  
                "city": {  
                    "type": "string",  
                    "description": "Name of the city to get weather for (e.g., 'New York', 'London')  
                }  
            },  
            "required": ["city"]  
        }  
    """;  
  
    // Create the tool with String schema  
    McpSchema.Tool weatherTool = new McpSchema.Tool(  
        name:"get_weather",  
        description:"Get current weather information for a specified city. Returns temperature data  
        schemaString  
    );
```



# Declaring our Tool for Weather

```
// Create the tool with String schema
McpSchema.Tool weatherTool = new McpSchema.Tool(
    name:"get_weather",
    description:"Get current weather information for a specified city. Returns temperature data for the requested location",
    schemaString
);

McpServerFeatures.SyncToolSpecification weatherToolSpecification = new McpServerFeatures.SyncToolSpecification(
    weatherTool,
    (exchange, request) -> {
        // Tool implementation – extract city and get weather data
        String city = extractCityFromArguments(request);
        String weatherData = getWeatherData(city);

        // Return the weather data as text content
        List<McpSchema.Content> contents = new ArrayList<>();
        contents.add(new McpSchema.TextContent(weatherData));

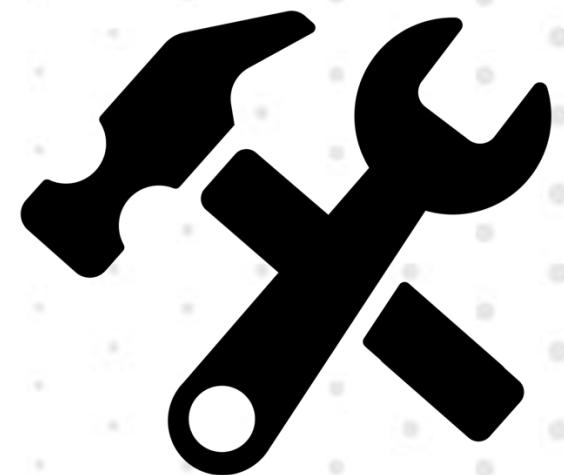
        return new McpSchema.CallToolResult(contents, isError:false);
    }
);

return weatherToolSpecification;
```

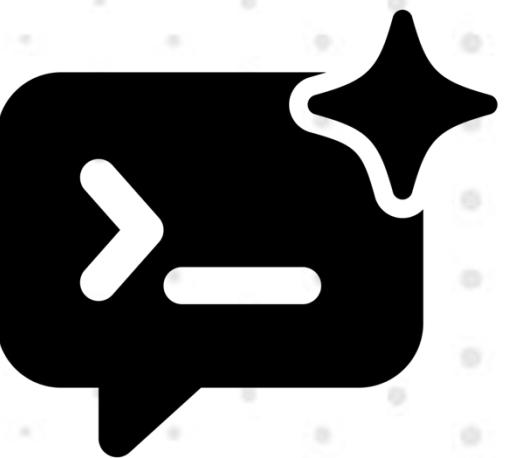


# Declaring our Tool for Weather

```
private static String getWeatherData(String city) {  
    // Get current timestamp for the response  
    String timestamp = LocalDateTime.now().format(  
        DateTimeFormatter.ofPattern(pattern:"yyyy-MM-dd HH:mm:ss"))  
};  
  
// Create formatted weather response  
// NOTE: This is hardcoded for now – will be replaced with actual API calls  
String weatherReport = String.format(  
    "Weather Report for %s\n" +  
    "=====\\n" +  
    "Current Temperature: 84F\\n" +  
    "Conditions: Clear\\n" +  
    "Humidity: 65%%\\n" +  
    "Wind: Light breeze\\n" +  
    "Last Updated: %s\\n" +  
    "\\n" +  
    "Note: This is sample data. Weather API integration coming soon!",  
    city,  
    timestamp  
);  
  
// Log the response for debugging  
System.err.println("Returning weather data for " + city);  
  
return weatherReport;
```



# Declaring our Prompts



# Declaring our Prompts

```
private static List<McpServerFeatures.SyncPromptSpecification> createWeatherPromptSpecifications() {
    return List.of(
        // Weather inquiry prompt – helps Claude ask about weather
        new McpServerFeatures.SyncPromptSpecification(
            new McpSchema.Prompt(
                name:"weather_inquiry",
                description:"Template for asking about weather conditions in a specific location",
                List.of(new McpSchema.PromptArgument(name:"location", description:"The city or location to inquire about")),
                (exchange, request) -> {
                    Map<String, Object> args = request.arguments();
                    String location = args.get(key:"location").toString();

                    // Create a structured prompt for weather inquiries
                    String promptText = String.format(
                        "I need current weather information for %s. " +
                        "Please provide the temperature and any relevant weather conditions. " +
                        "If you need to use a tool to get this information, please do so.",
                        location
                    );

                    return new McpSchema.GetPromptResult(
                        "Weather Inquiry for " + location,
                        List.of(new McpSchema.PromptMessage(
                            McpSchema.Role.USER,
                            new McpSchema.TextContent(promptText)
                        ))
                    );
                }
            )
        )
    );
}
```



# Declaring our Prompts

```
// Weather travel advice prompt - helps with travel planning based on weather
new McpServerFeatures.SyncPromptSpecification(
    new McpSchema.Prompt(
        name:"weather_travel_advice",
        description:"Template for getting weather-based travel advice for a destination",
        List.of(
            new McpSchema.PromptArgument(name:"destination", description:"Travel destination city", required:true),
            new McpSchema.PromptArgument(name:"travel_date", description:"Planned travel date (optional)", required:false)
        ),
        (exchange, request) -> {
            Map<String, Object> args = request.arguments();
            String destination = args.get(key:"destination").toString();
            Object travelDate = args.get(key:"travel_date");

            String dateInfo = (travelDate != null) ?
                " for travel on " + travelDate.toString() :
                " for current conditions";

            String promptText = String.format(
                "I'm planning to travel to %s%s. " +
                "Please check the current weather conditions and provide advice on " +
                "what to pack and any weather-related considerations for my trip. " +
                "Use the weather tool to get current temperature data.",
                destination, dateInfo
            );
        }
);
```



## DEMO 4

## DEMO 4

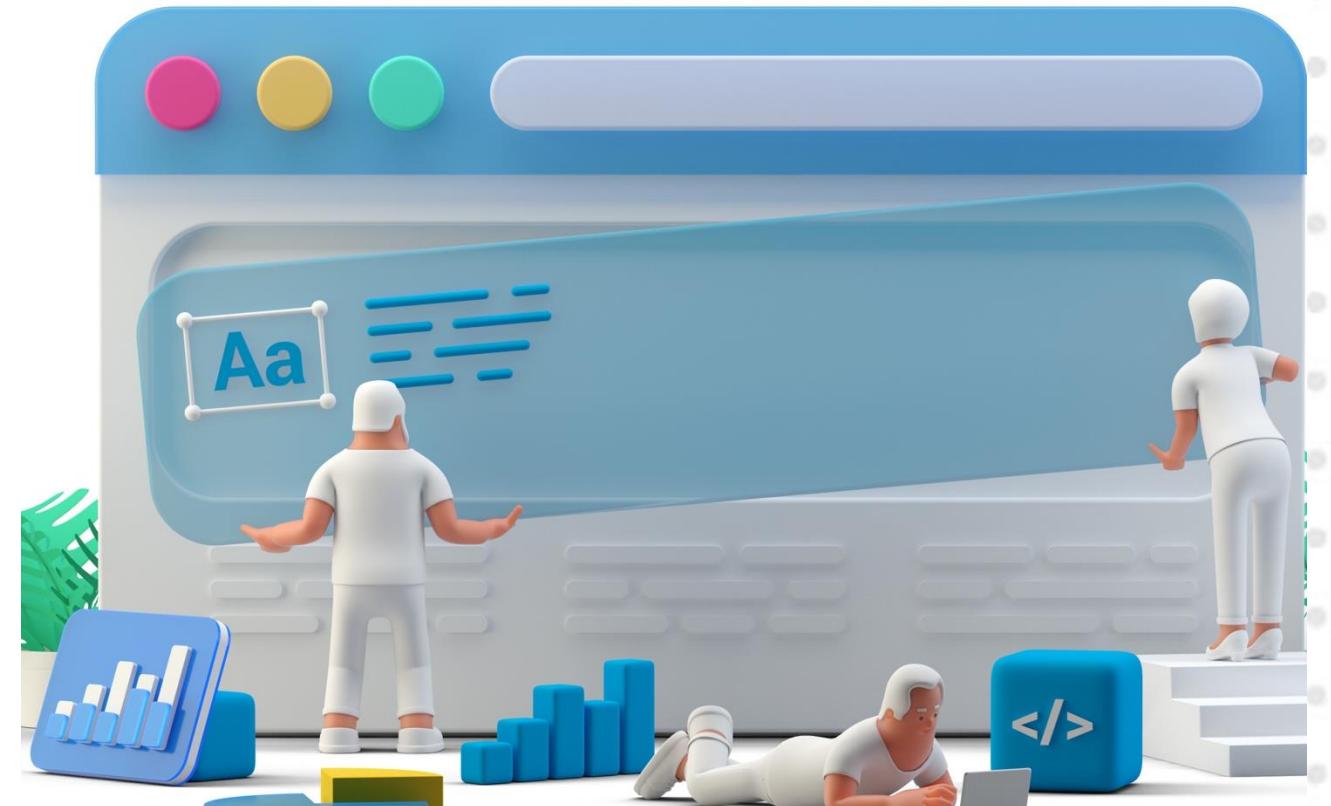


Postman



# Postman - How to Add a New MCP Server

The screenshot shows the Postman application interface. In the top navigation bar, there are links for Home, Workspaces, API Network, and a search bar labeled "Search Postman". On the far right of the top bar are buttons for "Invite", "Upgrade", and user profile. The main workspace is titled "MCP 1". On the left, a sidebar has tabs for Collections, Environments, Flows, and History. Under "Collections", there is a "New" button and an "Import" button. Below these are sections for "MCP Collection" and "Java STUDIO server". The "Java STUDIO server" section contains a command-line interface (CLI) field with the command "java -jar /Users/whitebeard/Desktop/mcpcode/weather-mcp-server/target/weather-mcp-server-1.0.0.jar". To the right of the CLI is a "Run" button. Below the CLI are tabs for "Message", "Environment", and "Settings", with "Message" being the active tab. Under "Tools", there are "Prompts" and "Resources" sections. A message card is displayed: "1. get\_weather" with the description "Get current weather information for a specified city. Returns temperature data for the requested location.". At the bottom of the main area, there are "Response" and "Notifications" tabs, a search bar, and a timestamp "17:48:55.346". A small icon of a character holding a wrench is in the bottom right. At the very bottom of the screen are various status icons and a toolbar with buttons for Postbot, Runner, Start Proxy, Cookies, Vault, Trash, and Help.



Postman

## DEMO 4



### PROMPT:

*What's the weather today in Seattle?*



Postman



Weather MCP



# Questions?



# 7 min Break

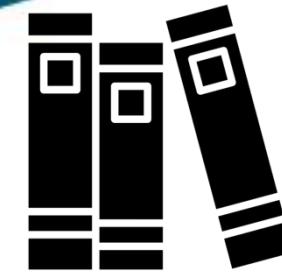


## PART 5

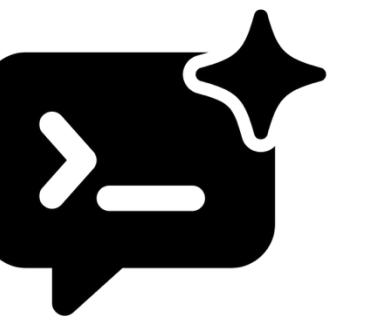
### Creating an MCP Server in Node.js



# Implementation Details



- Dependencies



- Prompts



- Transports



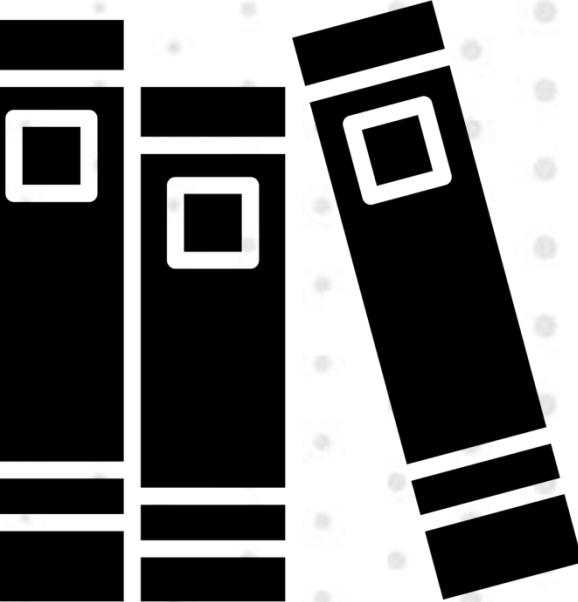
- Tools



# Dependencies

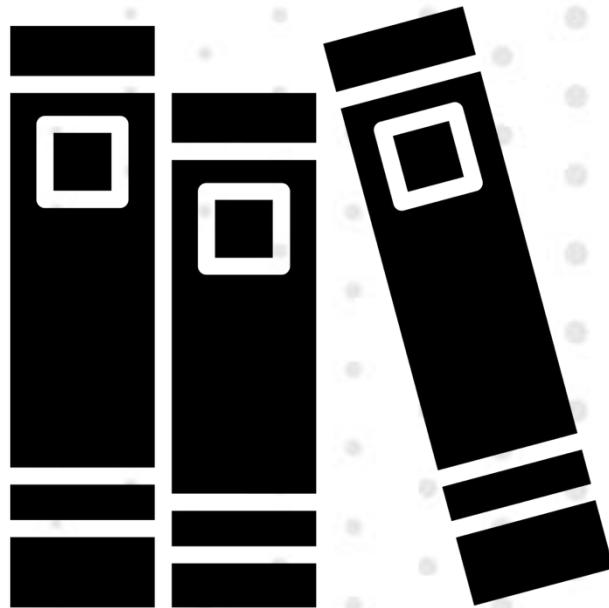
```
# Install MCP SDK  
npm install @modelcontextprotocol/sdk
```

```
# Install Zod for schema validation  
npm install zod
```



# Dependencies

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



# Create the MCP Server

```
// Create the MCP server
const server = new McpServer({
  name: "weather-mcp-server-nodejs-local",
  version: "1.0.0"
});
```



# Setting the Transport

```
const transport = new StdioServerTransport();  
await server.connect(transport);
```

# Setting the Transport

```
const transport = new StdioServerTransport();
await server.connect(transport);
```

```
async function main(): Promise<void> {
  try {
    // Start the server with STDIO transport
    console.error("Starting Weather MCP Server...");
    console.error("Server ready to accept connections via STDIO");

    // Create transport and connect
    const transport = new StdioServerTransport();
    await server.connect(transport);

    console.error("Weather MCP Server connected and running");

  } catch (error) {
    // Log error to stderr (stdout is used for MCP communication)
    console.error(`Failed to start Weather MCP Server: ${error}`);
    if (error instanceof Error) {
      console.error(error.stack);
    }
    process.exit(1);
  }
}
```



# Declaring our Tool for Weather

server.tool(



# Declaring our Tool for Weather

server.tool(

```
server.tool(  
  "get_weather",  
  {  
    city: z.string().describe("Name of the city to get weather for (e.g., 'New York', 'London', 'Tokyo')")  
  },  
  async ({ city }) => {  
    // Validate city parameter  
    if (!city || !city.trim()) {  
      throw new Error("City parameter cannot be empty");  
    }  
  
    const cleanCity = city.trim();  
  
    // Log the request for debugging (to stderr to avoid interfering with STDO)  
    console.error(`Processing weather request for city: ${cleanCity}`);  
  
    // Get current timestamp for the response  
    const timestamp = new Date().toLocaleString();
```



# Declaring our Tool for Weather

```
// Get current timestamp for the response
const timestamp = new Date().toLocaleString();

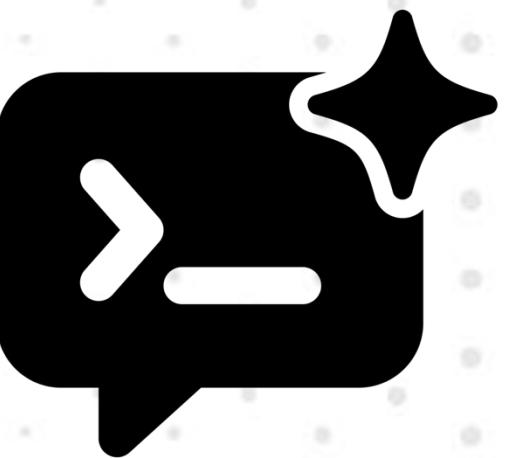
// Create formatted weather response
// NOTE: This is hardcoded for now – will be replaced with actual API calls
const weatherReport = `Weather Report for ${cleanCity}
=====
Current Temperature: 93F
Conditions: Clear
Humidity: 65%
Wind: Light breeze
Last Updated: ${timestamp}
`;

// Log the response for debugging
console.error(`Returning weather data for ${cleanCity}`);

return {
  content: [{ type: "text", text: weatherReport }]
};
```



# Declaring our Prompts



# Declaring our Prompts

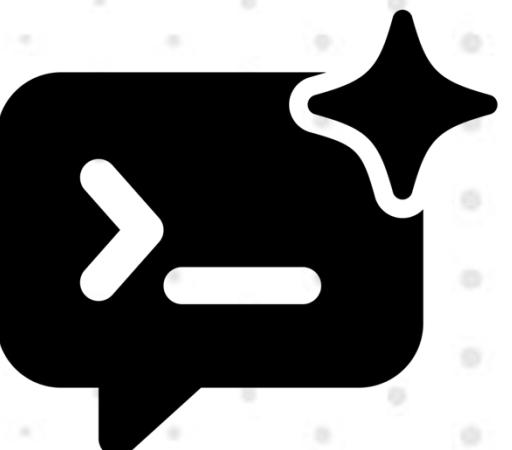
```
server.prompt(
```



# Declaring our Prompts

server.prompt(

```
server.prompt(  
  "weather_inquiry",  
  {  
    location: z.string().describe("The city or location to inquire about")  
  },  
  ({ location }) => {  
    const promptText = `I need current weather information for ${location}. ` +  
      "Please provide the temperature and any relevant weather conditions. " +  
      "If you need to use a tool to get this information, please do so."  
  
    return {  
      description: `Template for asking about weather conditions in ${location}`,  
      messages: [  
        {  
          role: "user",  
          content: {  
            type: "text",  
            text: promptText  
          }  
        }  
      ]  
    };  
  }  
);
```



# Declaring our Prompts

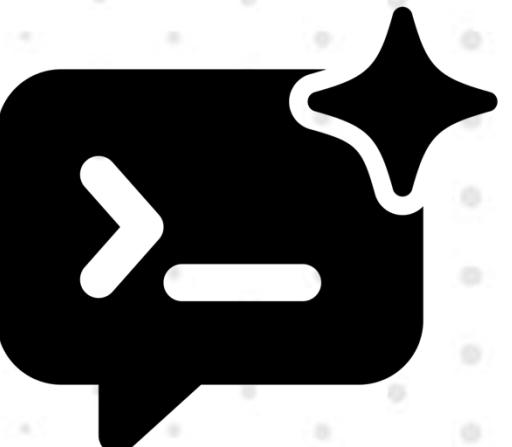
```
server.prompt(
  "weather_travel_advice",
  {
    destination: z.string().describe("Travel destination city"),
    travel_date: z.string().optional().describe("Planned travel date (optional)")
  },
  ({ destination, travel_date }) => {
    const dateInfo = travel_date ? ` for travel on ${travel_date}` : " for current conditions";

    const promptText = `I'm planning to travel to ${destination}${dateInfo}. ` +
      "Please check the current weather conditions and provide advice on " +
      "what to pack and any weather-related considerations for my trip. " +
      "Use the weather tool to get current temperature data.";

    return {
      description: `Template for getting weather-based travel advice for ${destination}`,
      messages: [
        {
          role: "user",
          content: {
            type: "text",
            text: promptText
          }
        }
      ]
    };
  }
);
```



Pe



## DEMO 5

# Weather MCP Server - Node.js

# DEMO 5



Claude Desktop



# Claude - How to Add a New MCP Server

Dallas Weather Forecast ▾

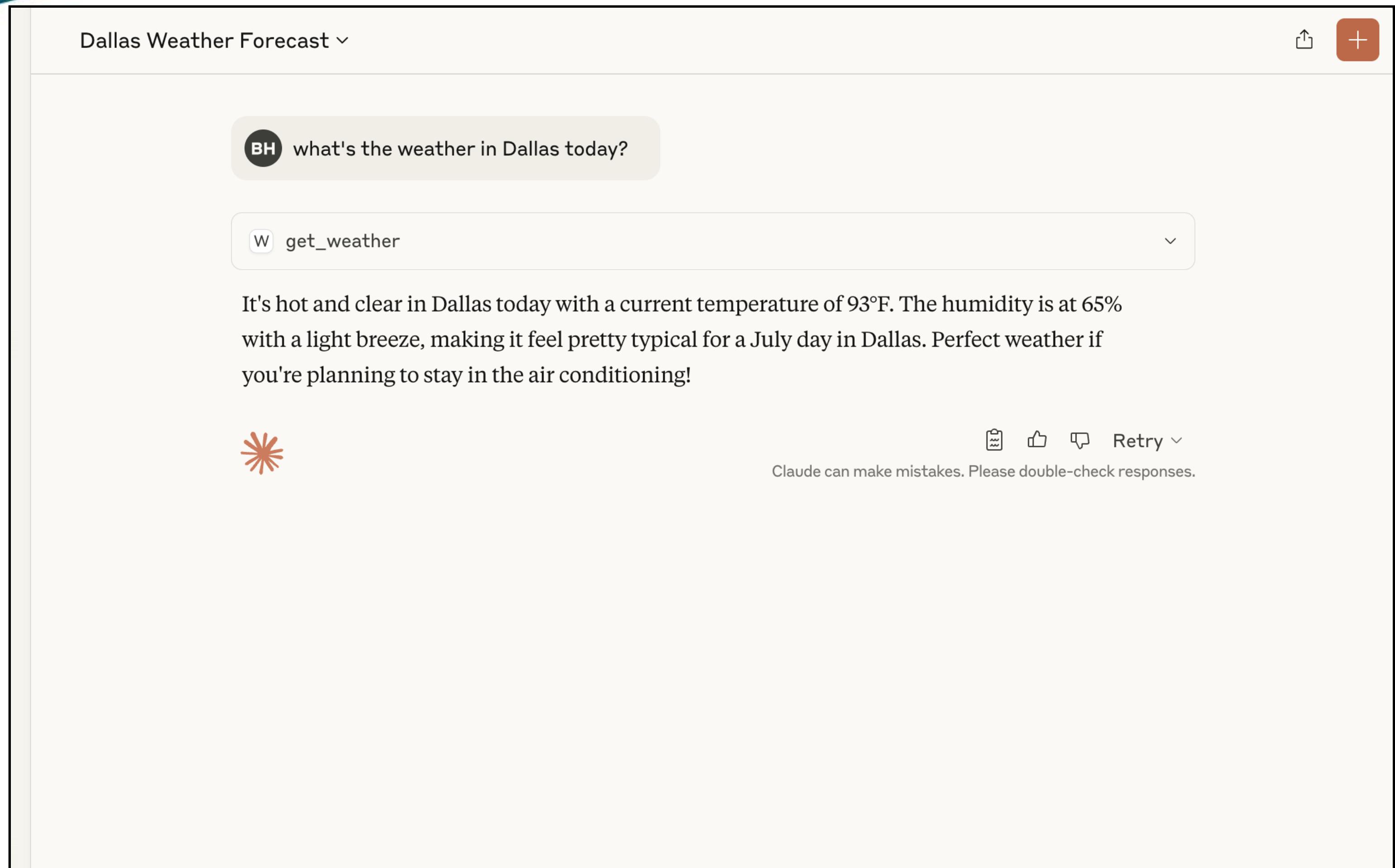
**BH** what's the weather in Dallas today?

**W** get\_weather

It's hot and clear in Dallas today with a current temperature of 93°F. The humidity is at 65% with a light breeze, making it feel pretty typical for a July day in Dallas. Perfect weather if you're planning to stay in the air conditioning!

✖️

Clipboard icon, Like icon, Dislike icon, Retry dropdown, Claude can make mistakes. Please double-check responses.



Claude Desktop

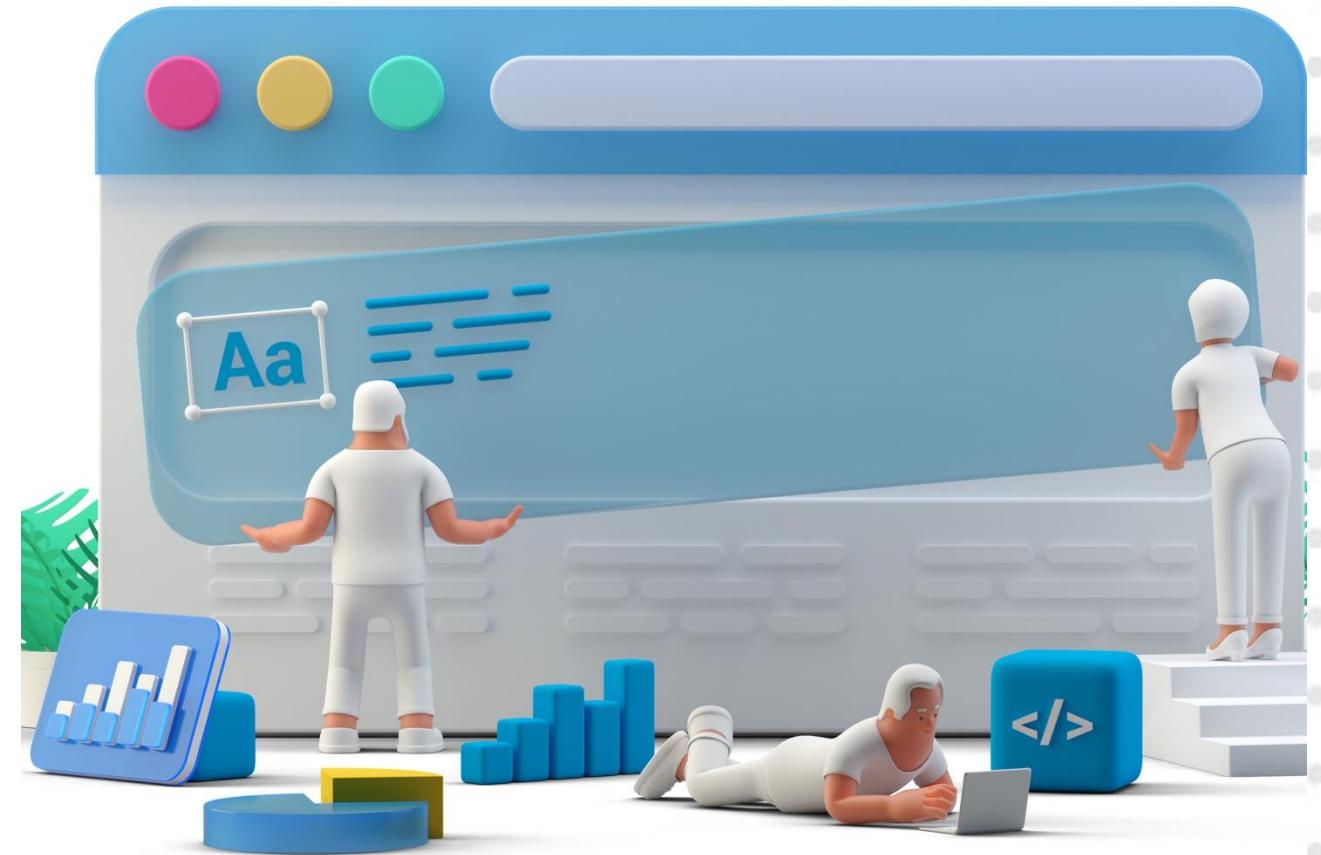
# Weather MCP Server - Node.js

## DEMO 5



### PROMPT:

*What's the weather today in Seattle?*



Claude Desktop



# Questions?



## PART 6

### Using MCP with the PostgreSQL Database

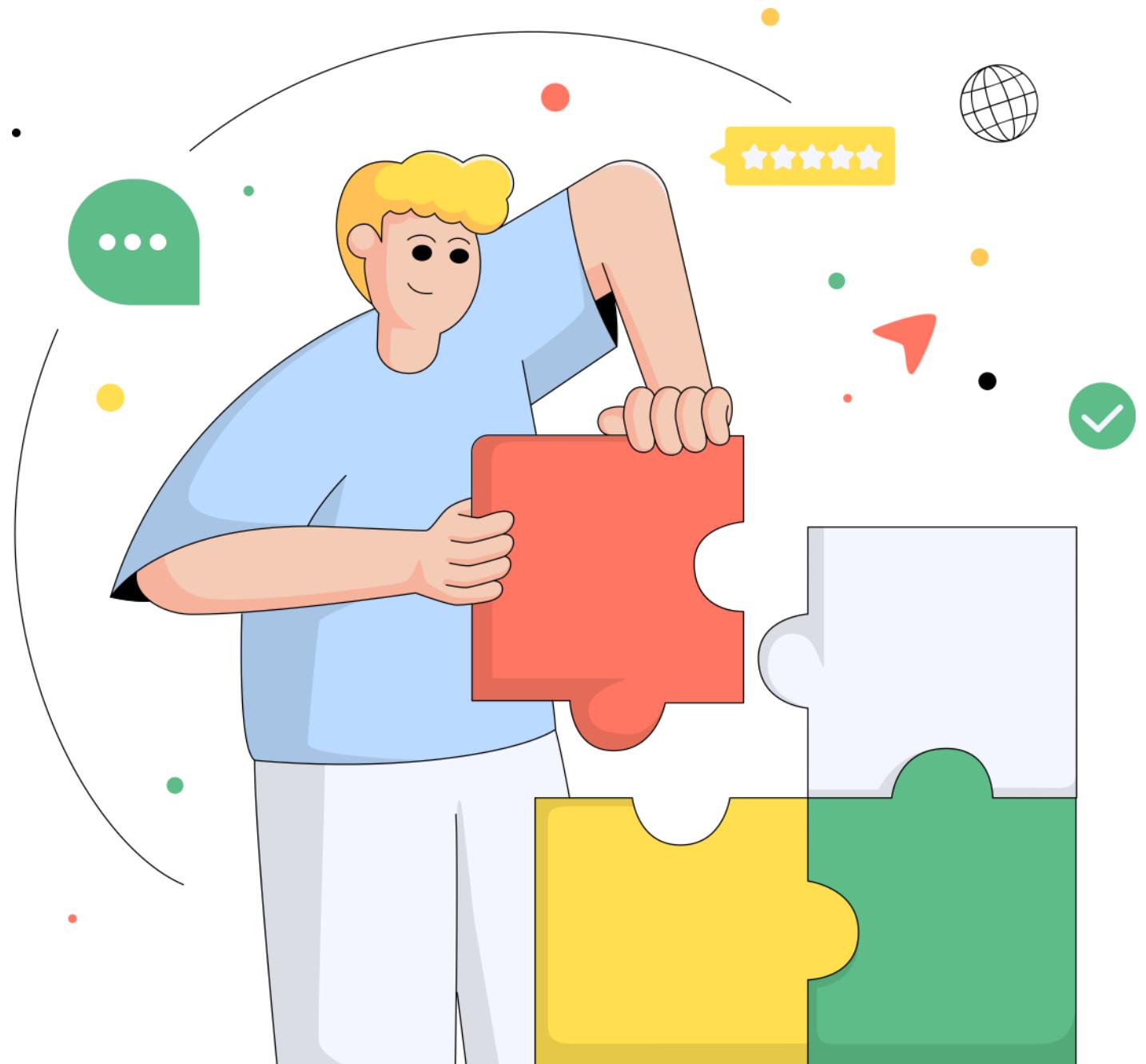


# What have we accomplished so far?

- We are very familiar configuring MCP clients
- We know how build MCP servers in 3 languages: Python, Java, Node.js



# What can be improved upon?



- One of the major points about MCP, is that we don't always have to build everything ourselves
- Let's learn how to accomplish things with one of the **700+** MCP servers available

# You Work for a Shoe Retailer

A red rectangular graphic featuring the word "RUNNING" in large, bold, black letters. A blue and orange running shoe is positioned diagonally across the letters, with blue liquid splashes trailing from its sole. Below the shoe, the word "Shoes" is written in a small, black, sans-serif font.

FEATURE PRODUCT

# Problem: Your CIO is Tired Custom SW Projects



- Let's talk about real numbers

# Problem: Your CIO is Tired Custom SW Projects



- Let's talk about real numbers
- **31.1% of internal software** projects are canceled before completion

# Problem: Your CIO is Tired Custom SW Projects



- Let's talk about real numbers
- **31.1% of internal software** projects are canceled before completion
- **52.7% exceed** their original budgets

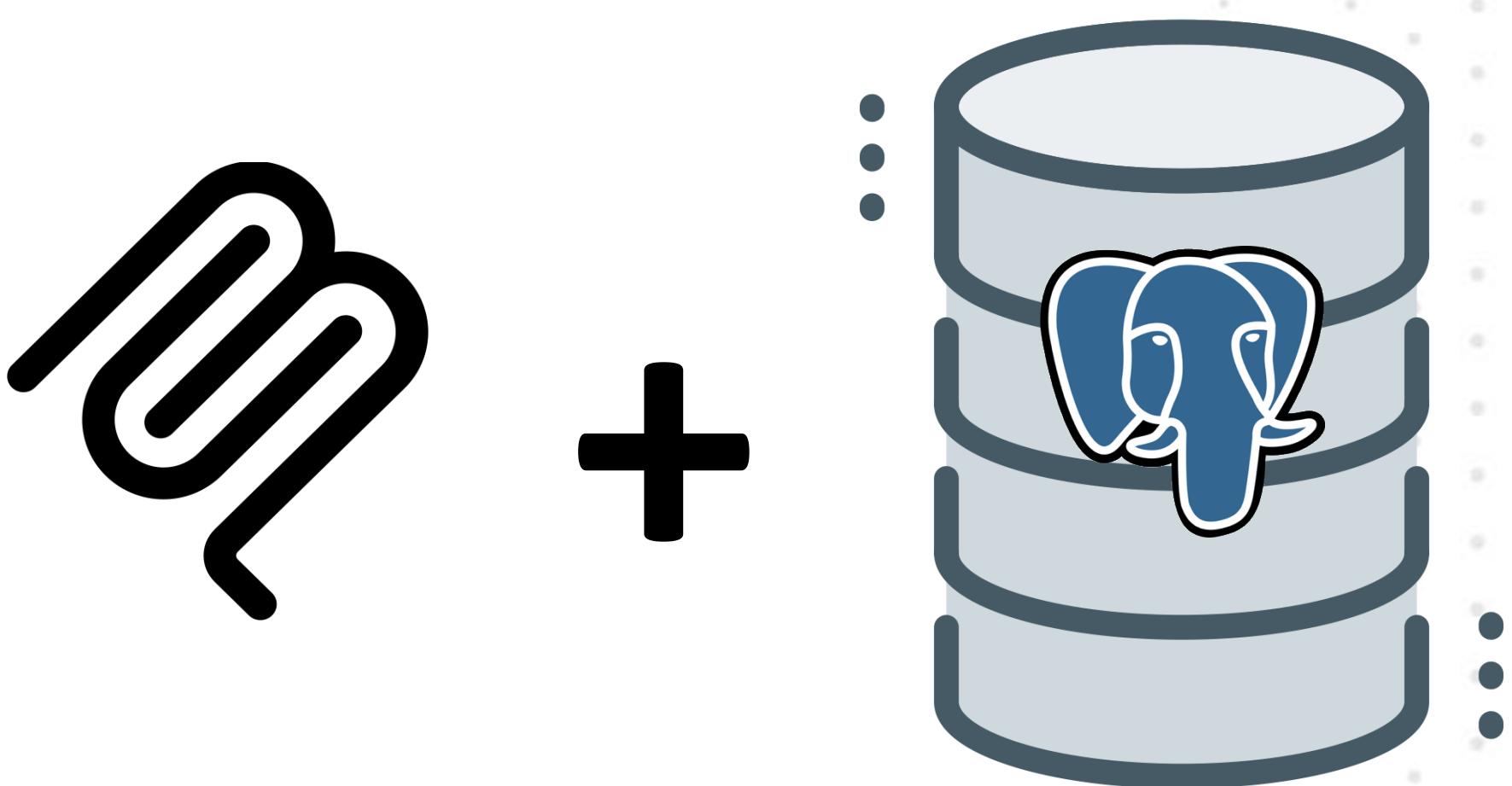
# Management Has a Question

**What % of  
customers shop  
online **vs** retail?**



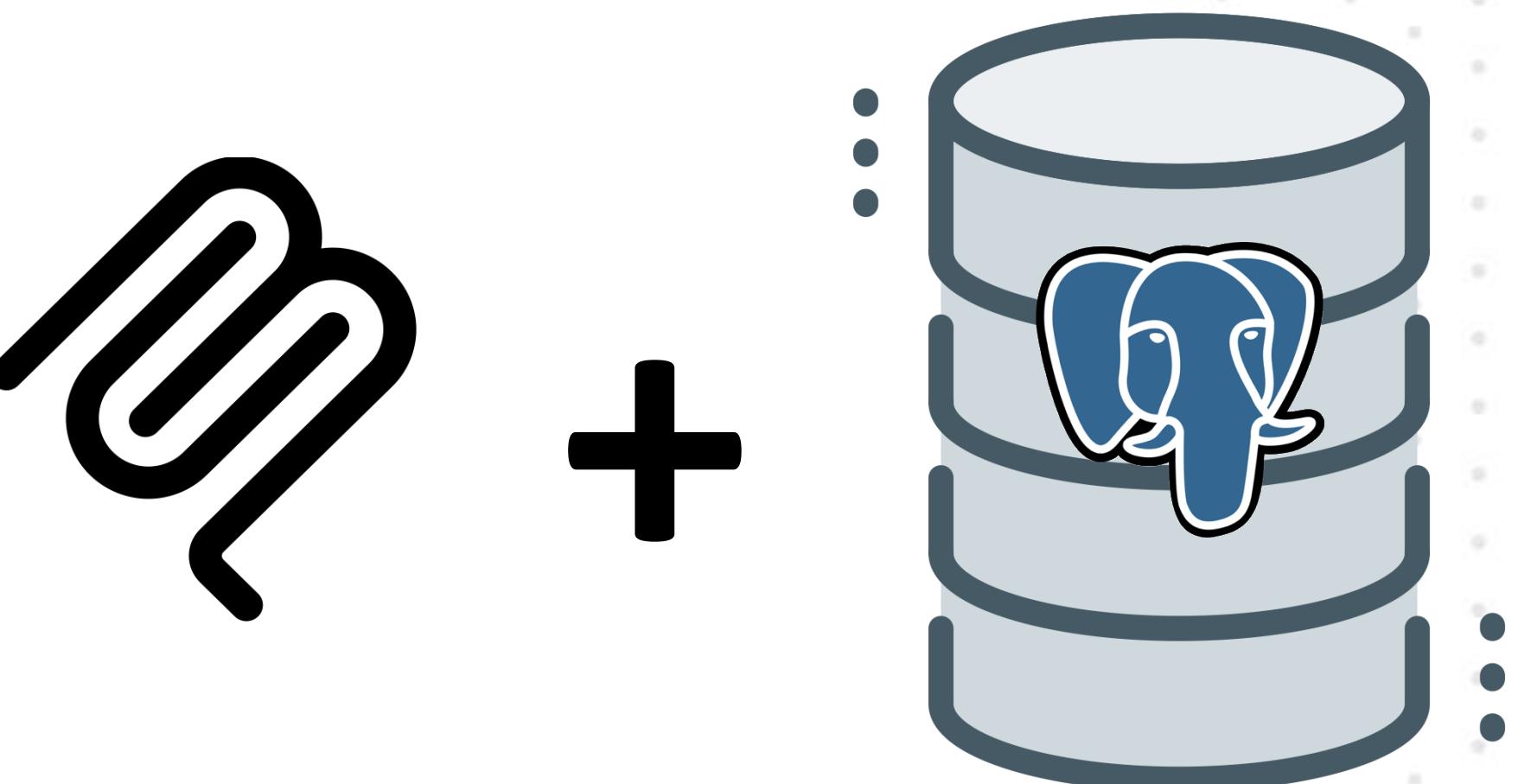
# Solution: Off the Shelf MCP Server

- Let's use an MCP Server to interface with our database



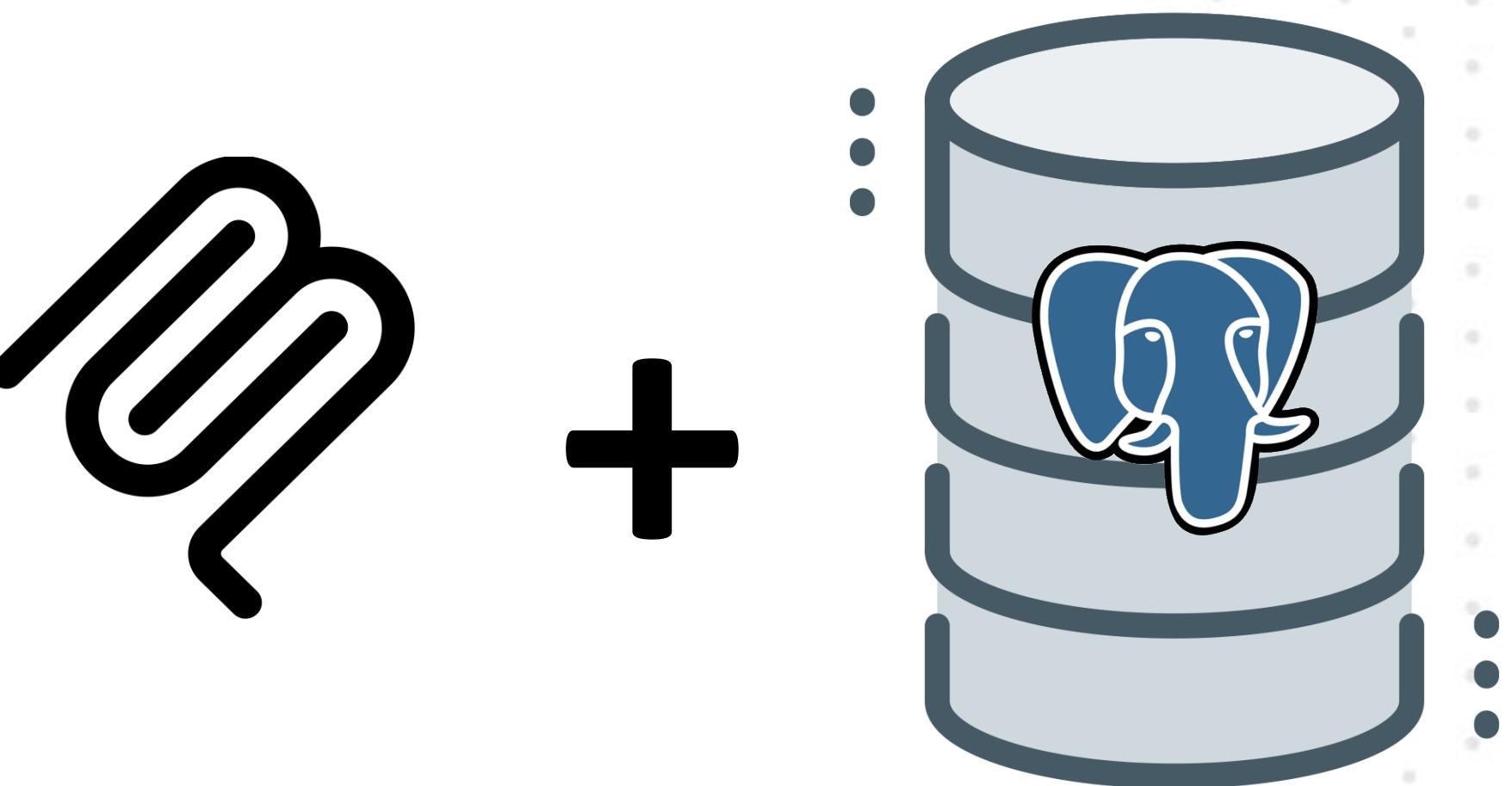
# Solution: Off the Shelf MCP Server

- Let's use an MCP Server to interface with our database
- Instead of building a custom application, let our managers talk to the DB themselves



# Solution: Off the Shelf MCP Server

- Let's use an MCP Server to interface with our database
- Instead of building a custom application, let our managers talk to the DB themselves
- We have a few options

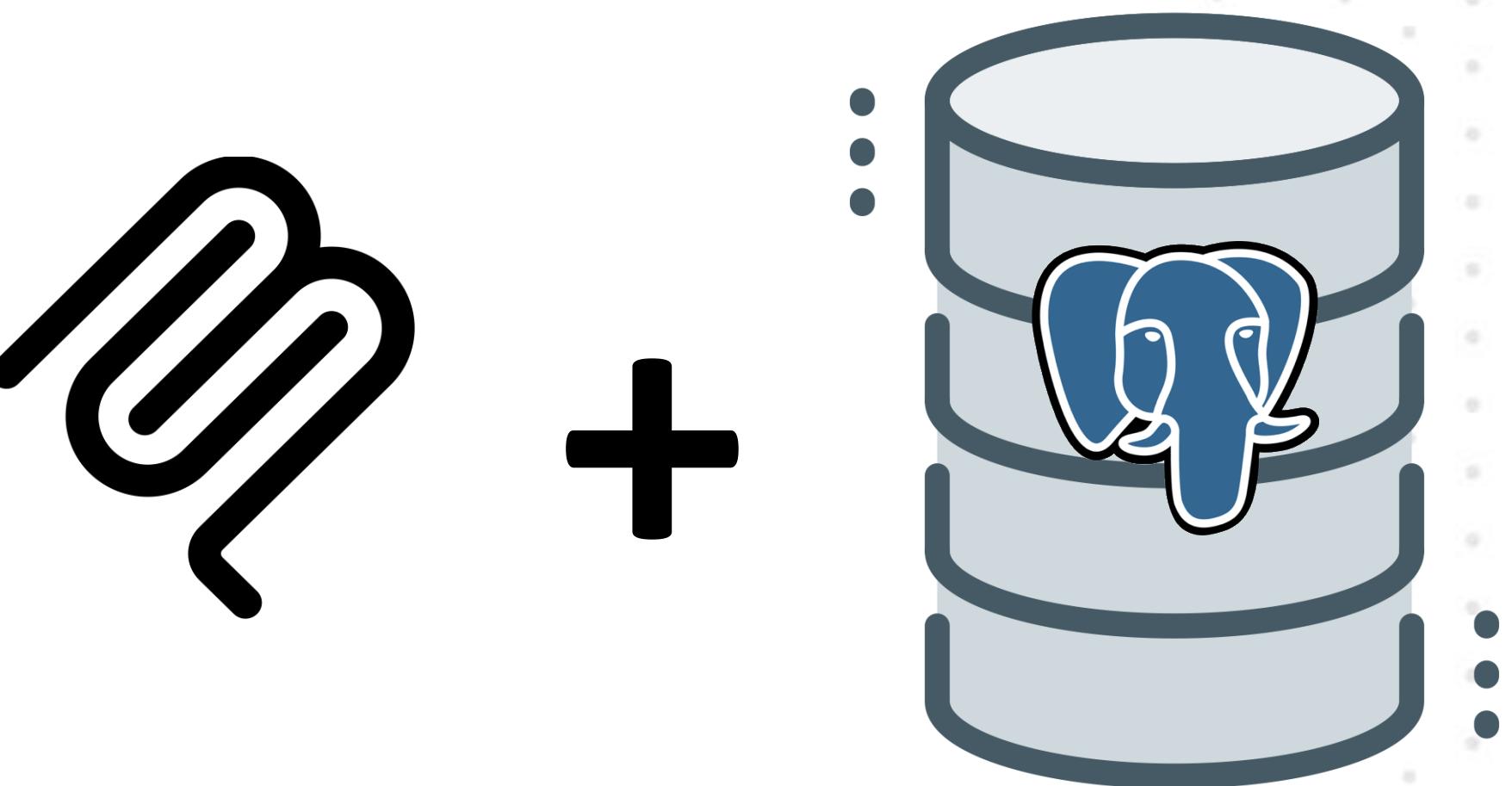


# PostgreSQL MCP Servers

- Original Anthropic MCP Server:

archived

<https://github.com/modelcontextprotocol/servers-archived/tree/main/src/postgres>



# PostgreSQL MCP Servers

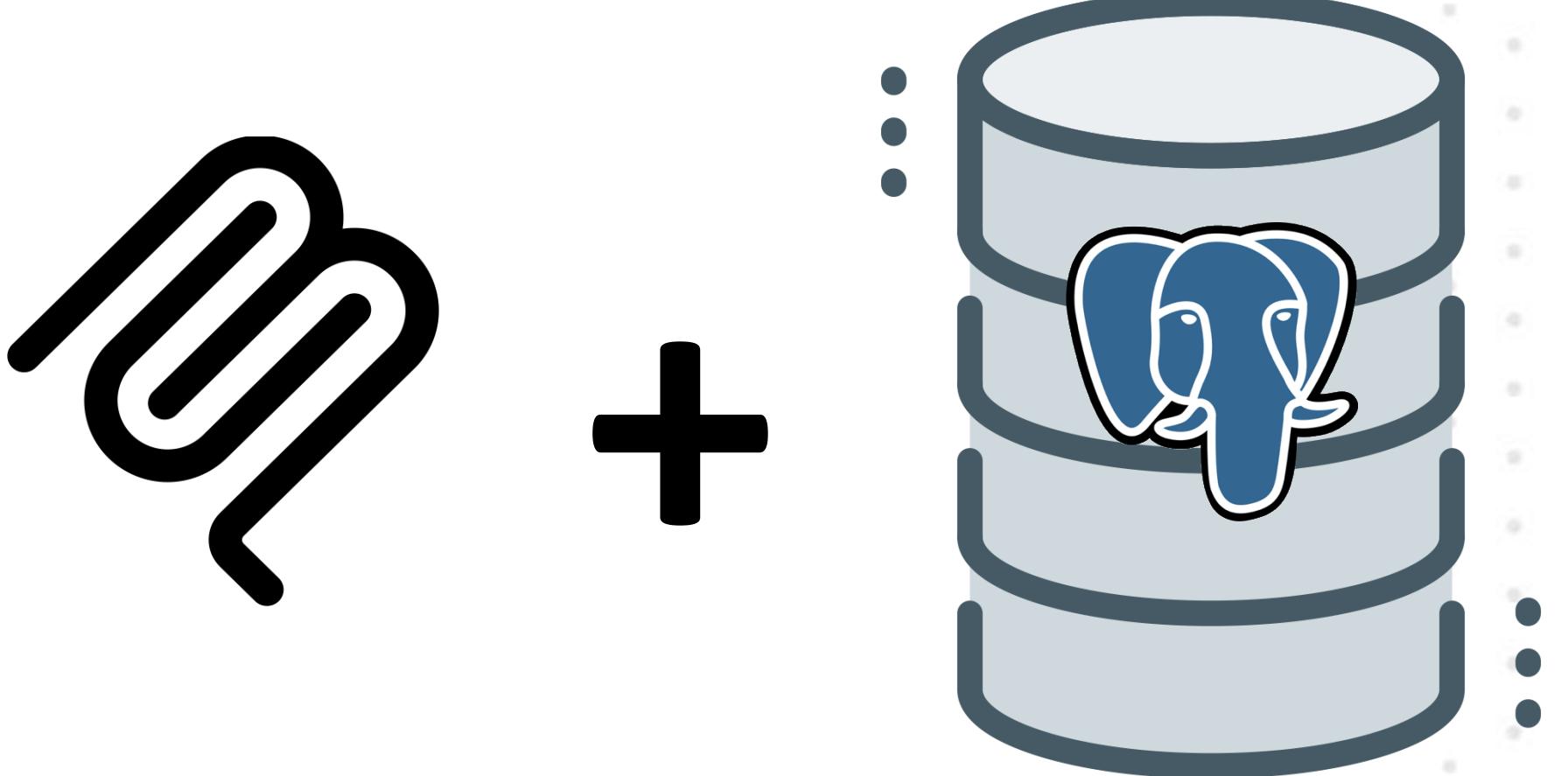
- Original Anthropic MCP Server:

archived

<https://github.com/modelcontextprotocol/servers-archived/tree/main/src/postgres>

- HenkDz:

<https://github.com/HenkDz/postgresql-mcp-server>



# PostgreSQL MCP Servers

- Original Anthropic MCP Server:

archived

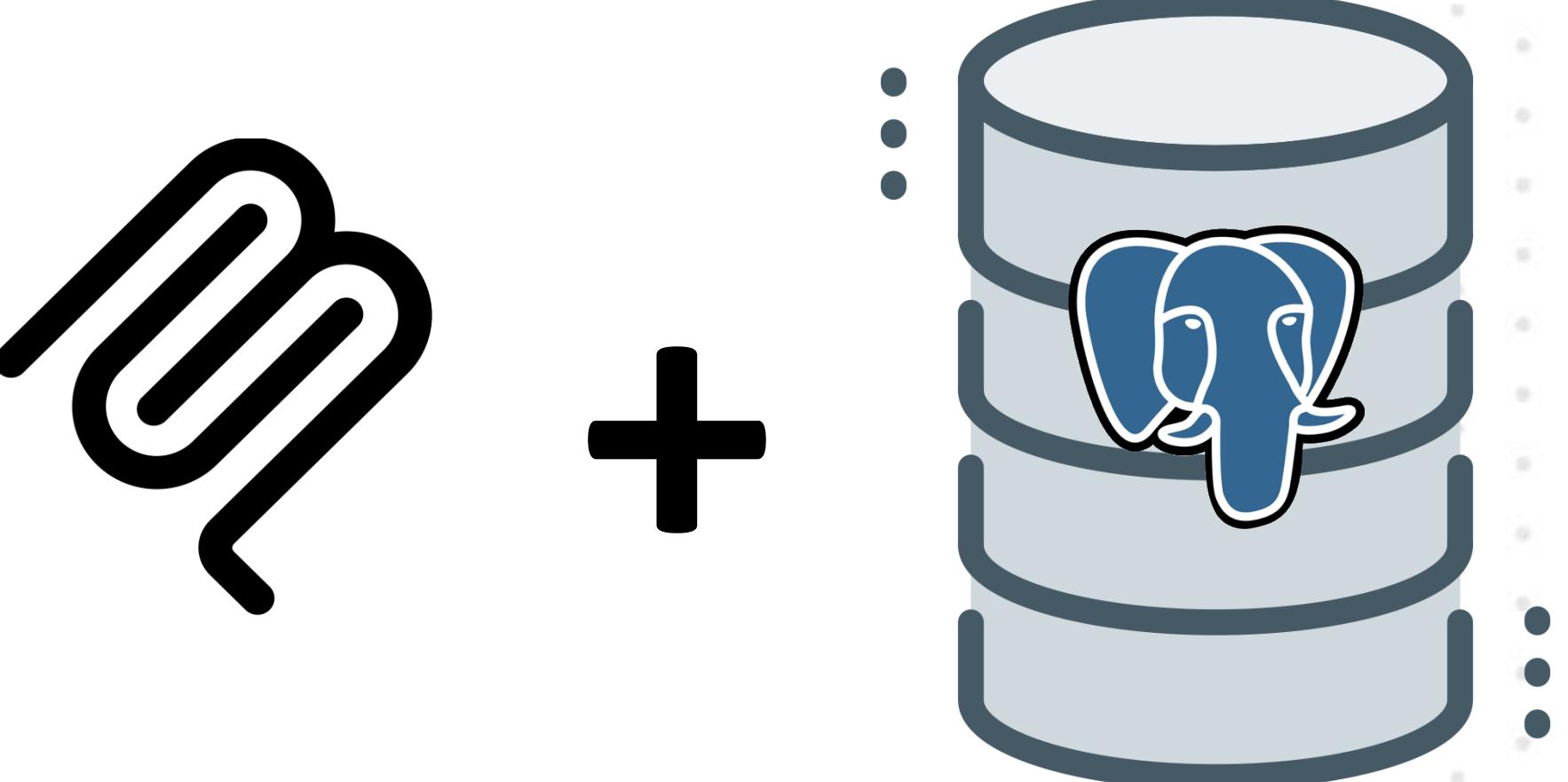
<https://github.com/modelcontextprotocol/servers-archived/tree/main/src/postgres>

- HenkDz:

<https://github.com/HenkDz/postgresql-mcp-server>

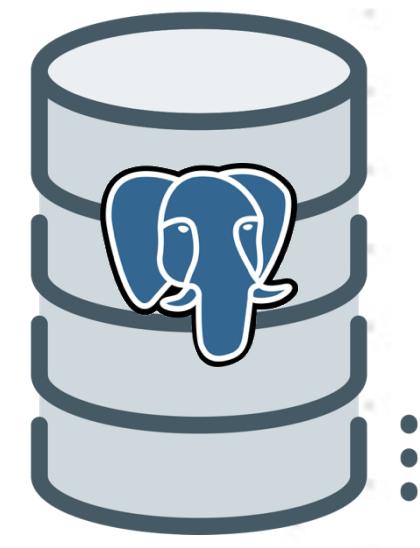
- Crystal DBA:

<https://github.com/crystaldba/postgres-mcp>



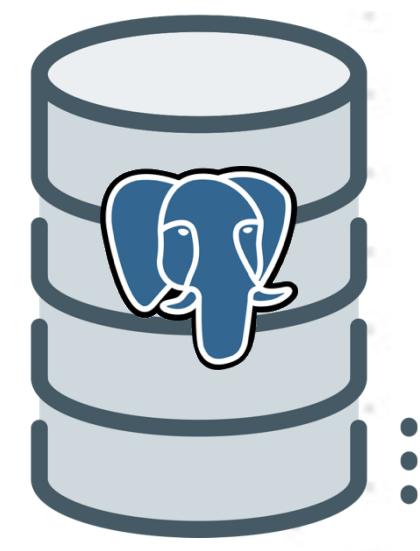
# How to Setup Your MCP Server with PostgreSQL

- Pre-requisites



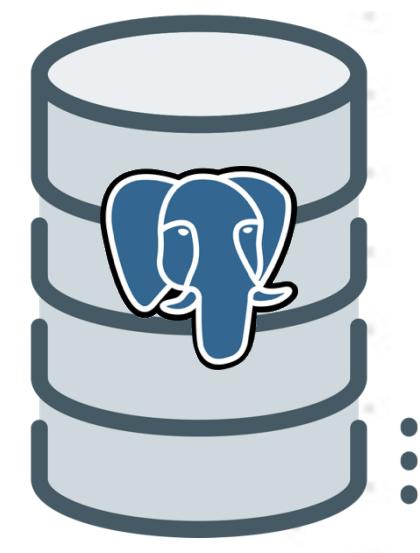
# How to Setup Your MCP Server with PostgreSQL

- Pre-requisites
- PostgreSQL DB  
(I'm using v 16)



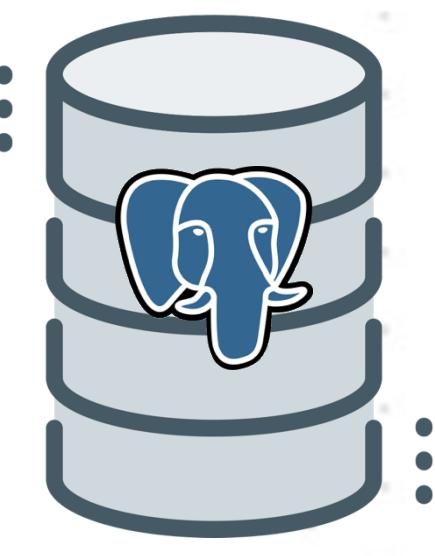
# How to Setup Your MCP Server with PostgreSQL

- Pre-requisites
- PostgreSQL DB  
(I'm using v 16)
- Node.js



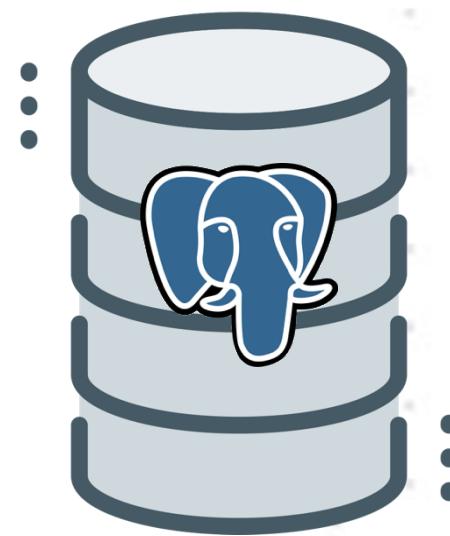
# How to Setup Your MCP Server with PostgreSQL

- Pre-requisites
- PostgreSQL DB  
(I'm using v 16)
- Node.js
- Claude Desktop

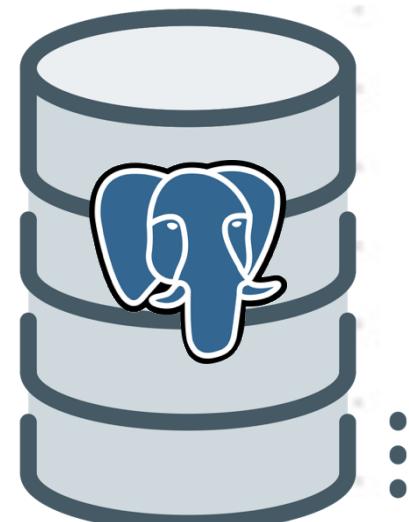
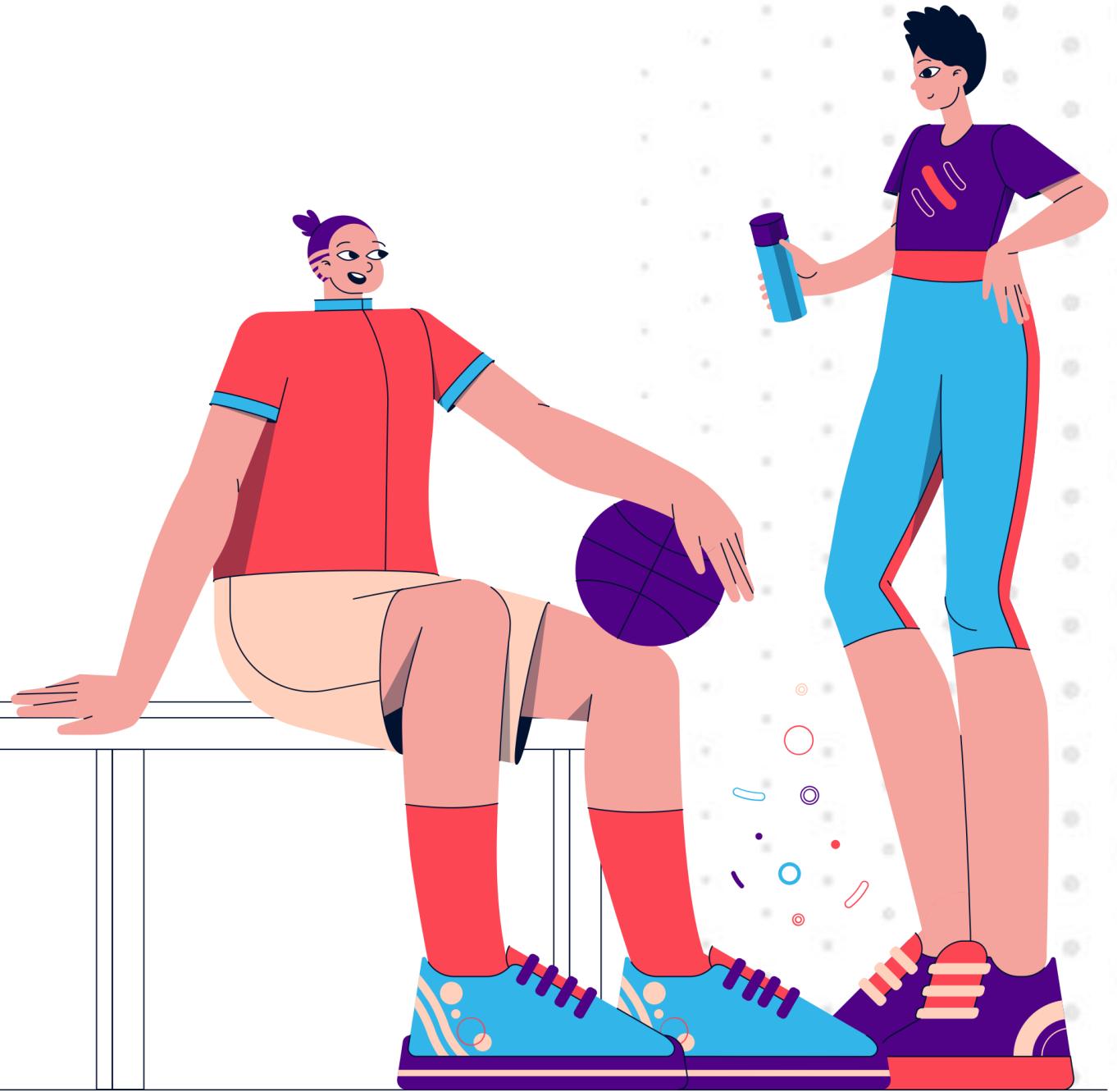


# Claude Configuration

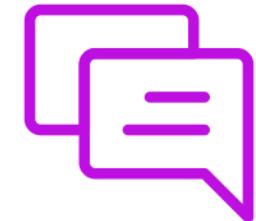
```
{  
  "mcpServers": {  
    "postgresql-mcp": {  
      "command": "npx",  
      "args": ["@henkey/postgres-mcp-server"],  
      "env": {  
        "POSTGRES_CONNECTION_STRING":  
          "postgresql://postgres:postgres@localhost:5432/shoe_store_analytics"  
      }  
    }  
  }  
}
```



# DEMO 6



# DEMO 6 - Prompts Used



## **PROMPT:**

1. *"How many customers prefer online shopping vs retail stores?"*
2. *"What's the total revenue from online sales compared to retail sales?"*
3. *"Which products are most popular in retail stores?"*
4. *"Show me customers who prefer online shopping but have made purchases in retail stores"*
5. *"What's the average order value for each purchase location?"*
6. *"Which Nike products have generated the most revenue?"*
7. *"How many orders were placed in the last 30 days?"*
8. *"What's the distribution of customers across different states?"*
9. *"Which retail store has the highest sales volume?"*
10. *"Show me the top 5 customers by total spending"*

# Ramifications



# Ramifications



- This will lead to a new generation of applications: NO User Interfaces

# Ramifications



- This will lead to a new generation of applications: NO User Interfaces
- The biggest impact: Internal corporate apps

# Ramifications



- This will lead to a new generation of applications: NO User Interfaces
- The biggest impact: Internal corporate apps
- The demo used PostgreSQL, but the technique can be applied to **any database**

# Questions?



## PART 7

### Course Wrap-up



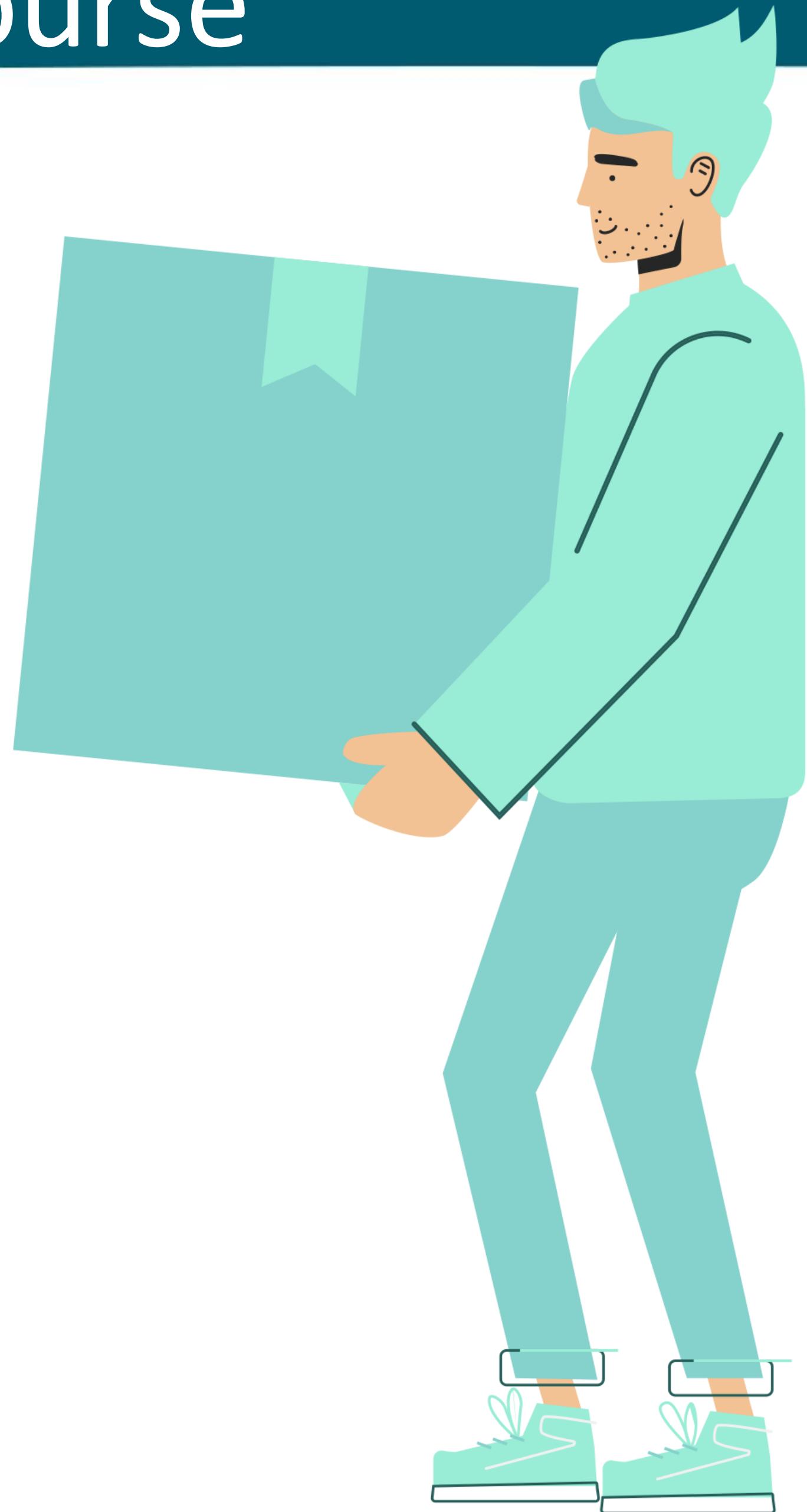
# Key Takeaways from This Course

## PART 1

You will **understand the basics** of the Model Context Protocol (MCP)

Understand the problems that it solves

You'll understand why the industry has given overwhelming support



# Who is Officially Supporting the MCP Standard?

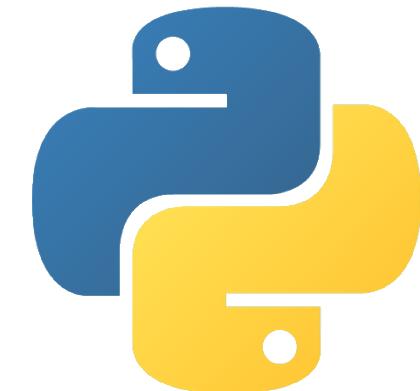


Microsoft  
Azure

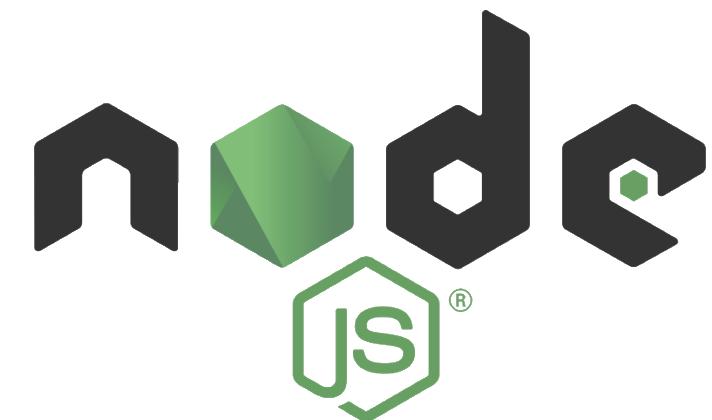
Gemini

Claude

# Massive Industry Partnerships



Python SDK

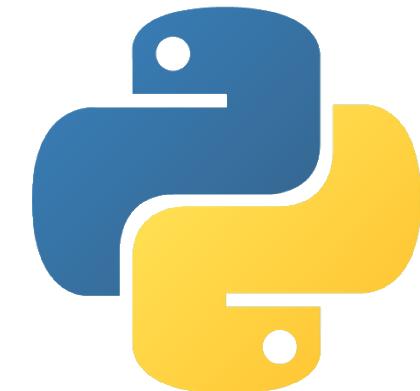


Node.js SDK

**ANTHROPIC**

**ANTHROPIC**

# Massive Industry Partnerships



Python SDK



Node.js SDK



Java SDK

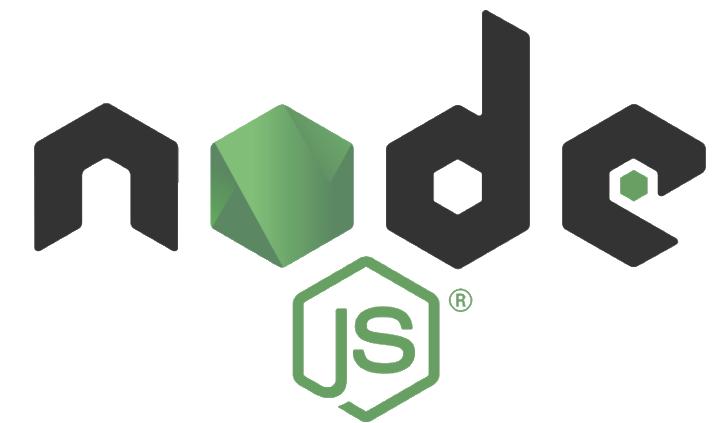
**ANTHROPIC**

**ANTHROPIC**

# Massive Industry Partnerships



Python SDK



Node.js SDK



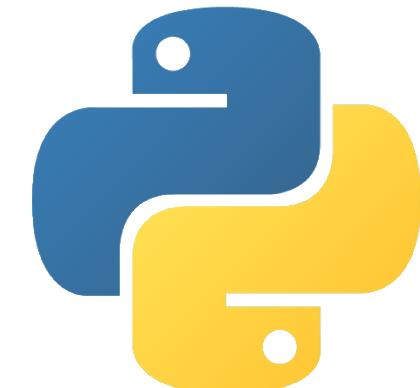
Java SDK

**ANTHROPIC**

**ANTHROPIC**



# Massive Industry Partnerships



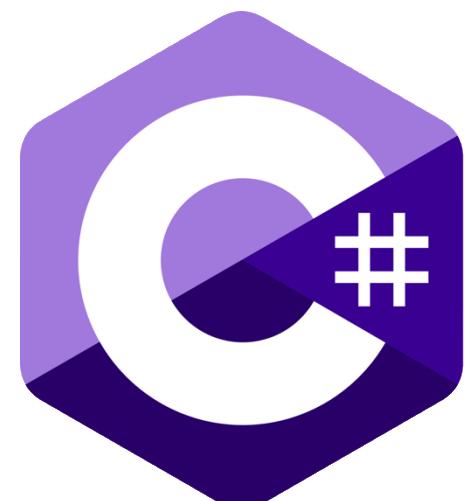
Python SDK



Node.js SDK



Java SDK



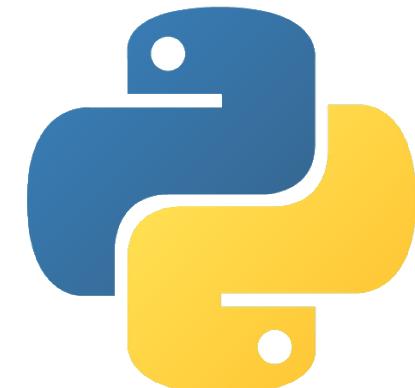
C# .NET SDK

**ANTHROPIC**

**ANTHROPIC**



# Massive Industry Partnerships



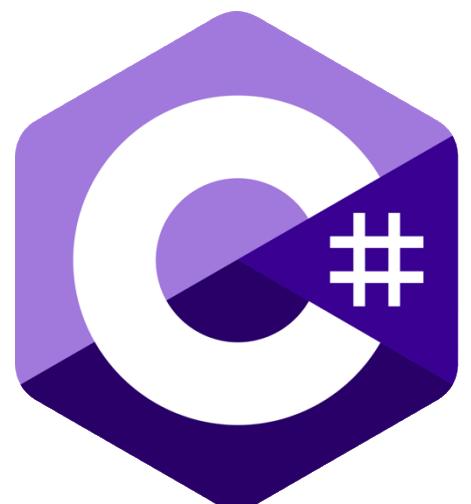
Python SDK



Node.js SDK



Java SDK



C# .NET SDK



**ANTHROPIC**

**ANTHROPIC**



# Massive Industry Partnerships



Ruby SDK

# Massive Industry Partnerships



Ruby SDK



*shopify*

# Massive Industry Partnerships



Ruby SDK



Go SDK



*shopify*

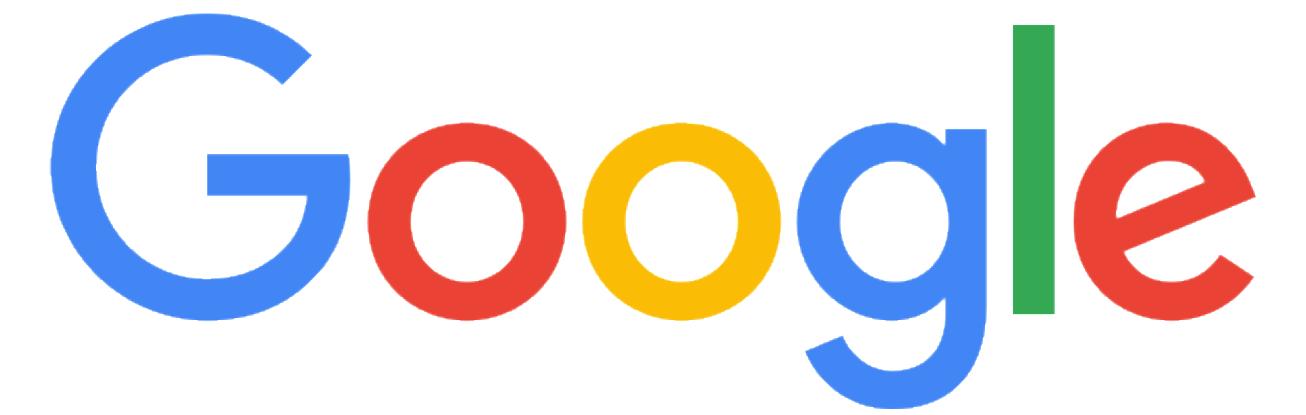
# Massive Industry Partnerships



Ruby SDK



Go SDK



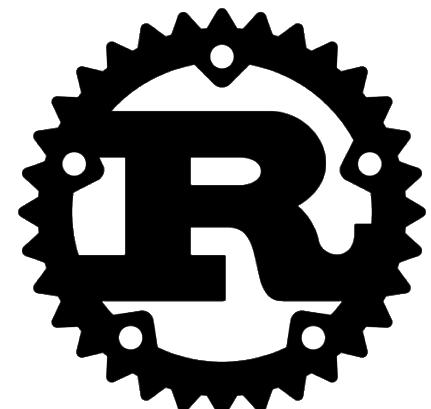
# Massive Industry Partnerships



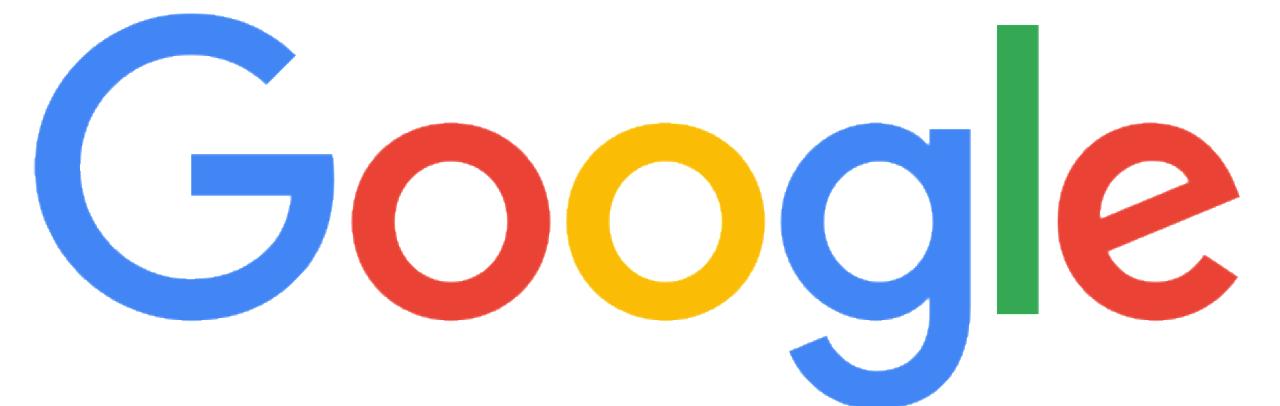
Ruby SDK



Go SDK



Rust SDK



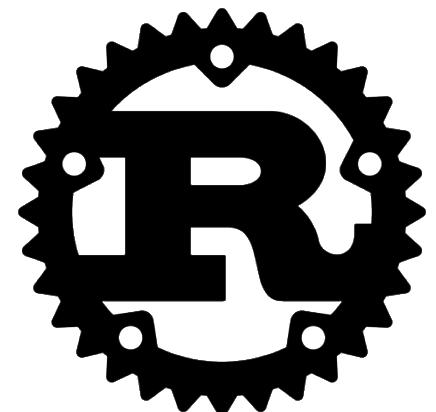
# Massive Industry Partnerships



Ruby SDK



Go SDK



Rust SDK



Google



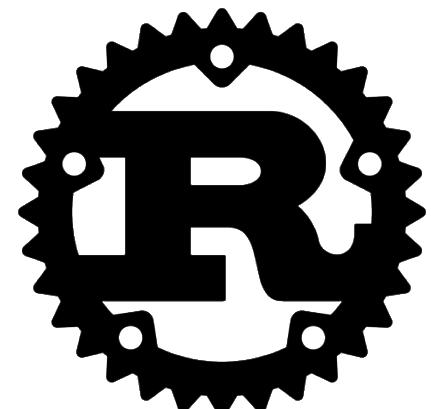
# Massive Industry Partnerships



Ruby SDK



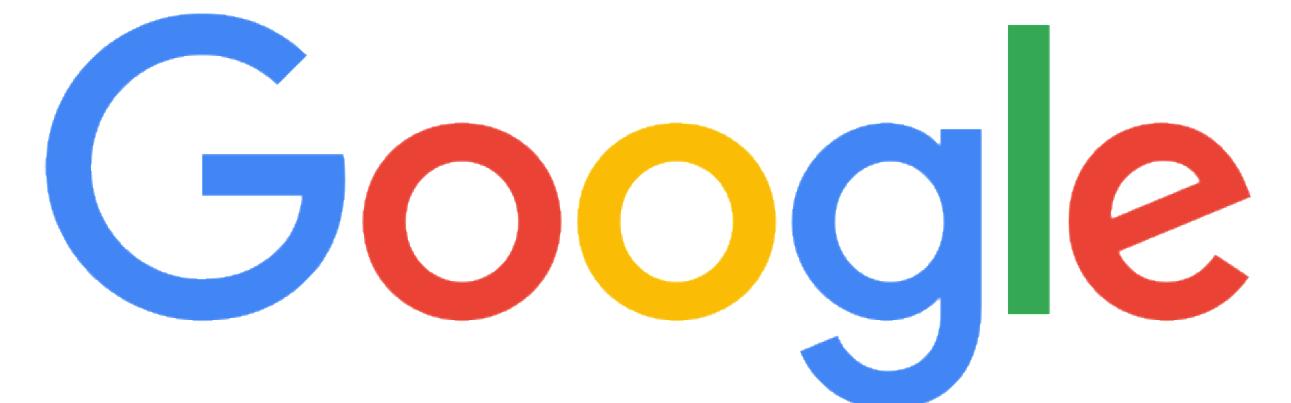
Go SDK



Rust SDK



Kotlin SDK



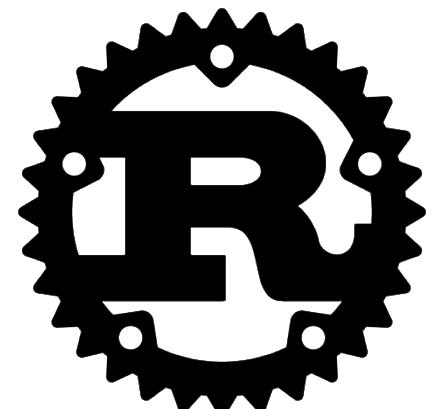
# Massive Industry Partnerships



Ruby SDK



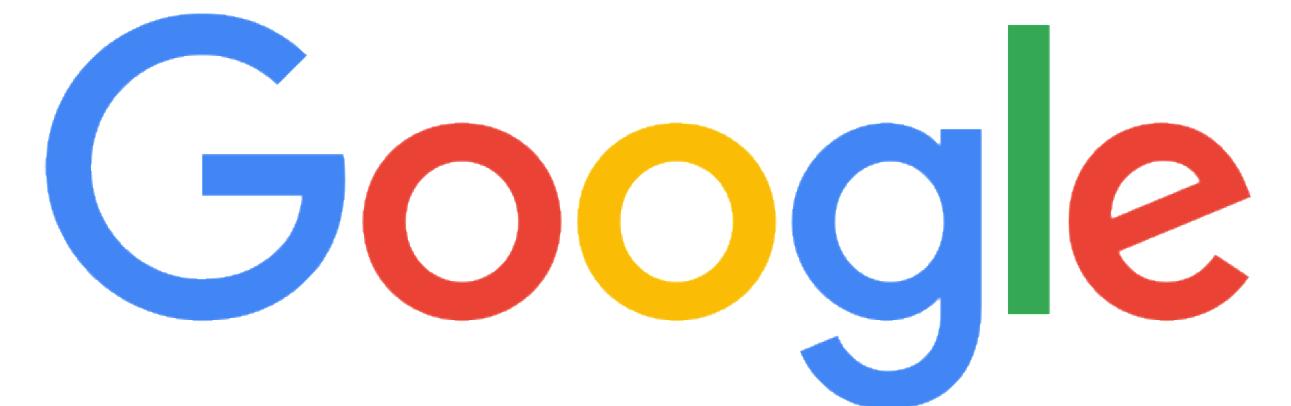
Go SDK



Rust SDK



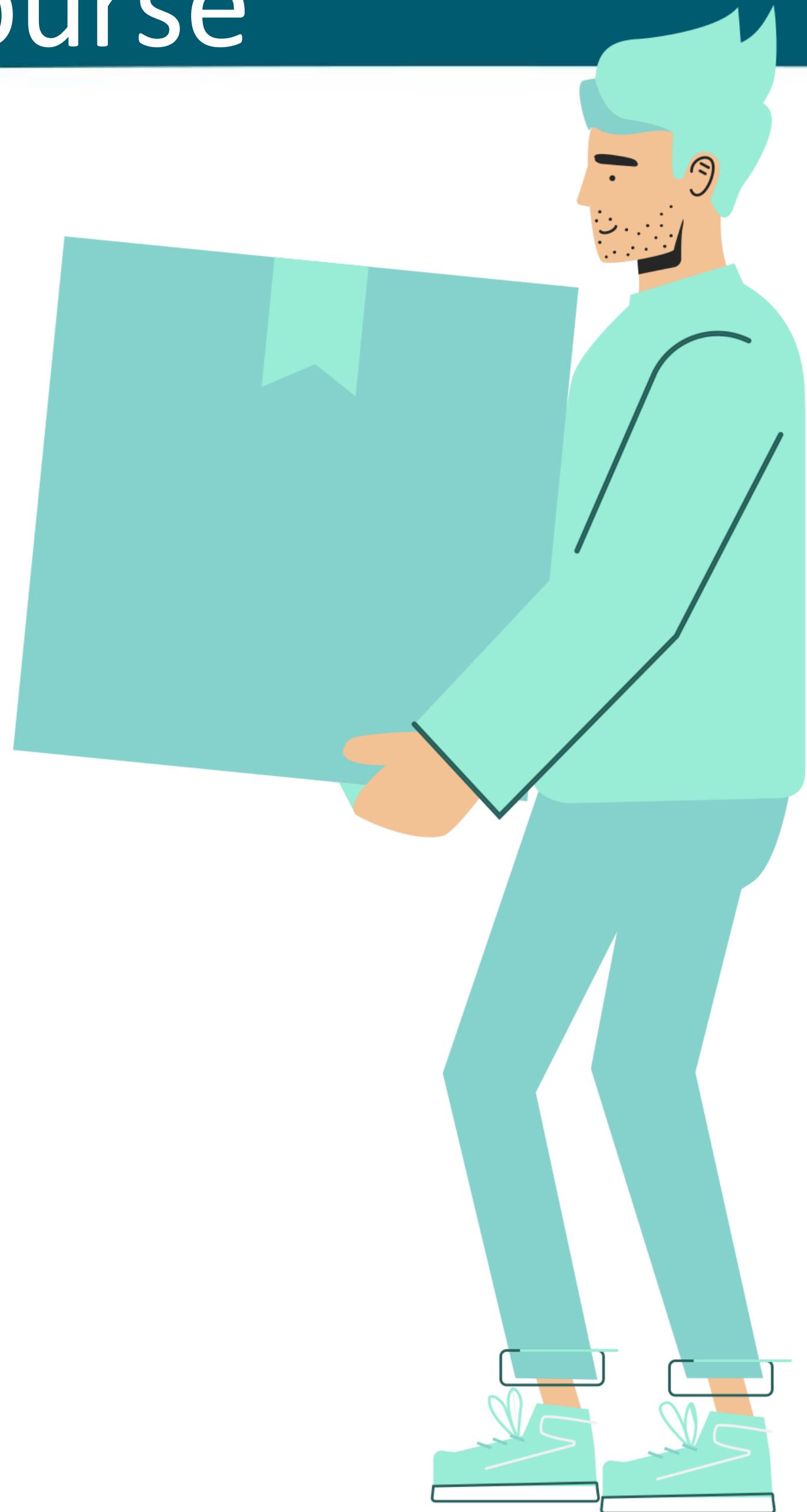
Kotlin SDK



# Key Takeaways from This Course

## PART 2

You take a deep dive into the MCP Protocol and learn the ins-and-outs of **how MCP Servers work**



# Here's the Point About MCP

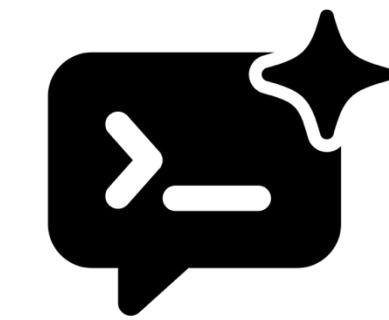
- It should work like HTTP
- Take any client that you want
- The client should not have any knowledge of server
- Access the MCP server that you want to perform the thing that you want to do



# What are the Basic Terms and Terminology?



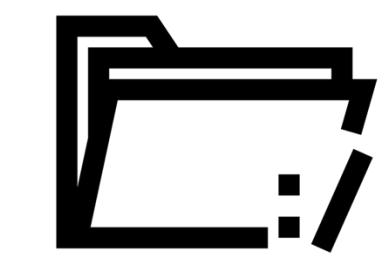
- Message Protocol



- Prompts



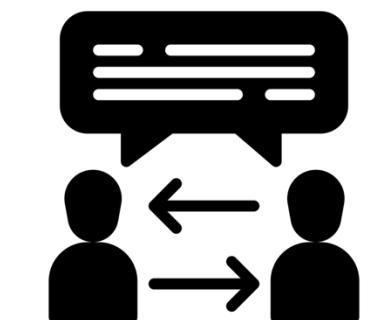
- Transports



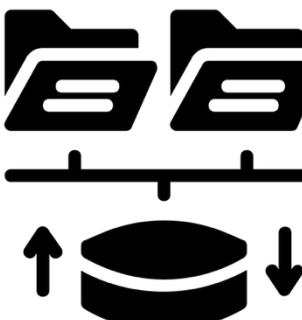
- Roots



- Tools



- Sampling



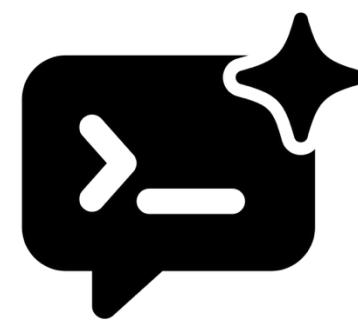
- Resources



# What are the Basic Terms and Terminology?



- Message Protocol



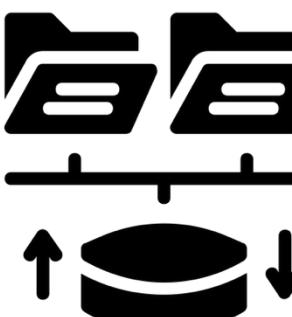
- Prompts



- Transports



- Tools



- Resources

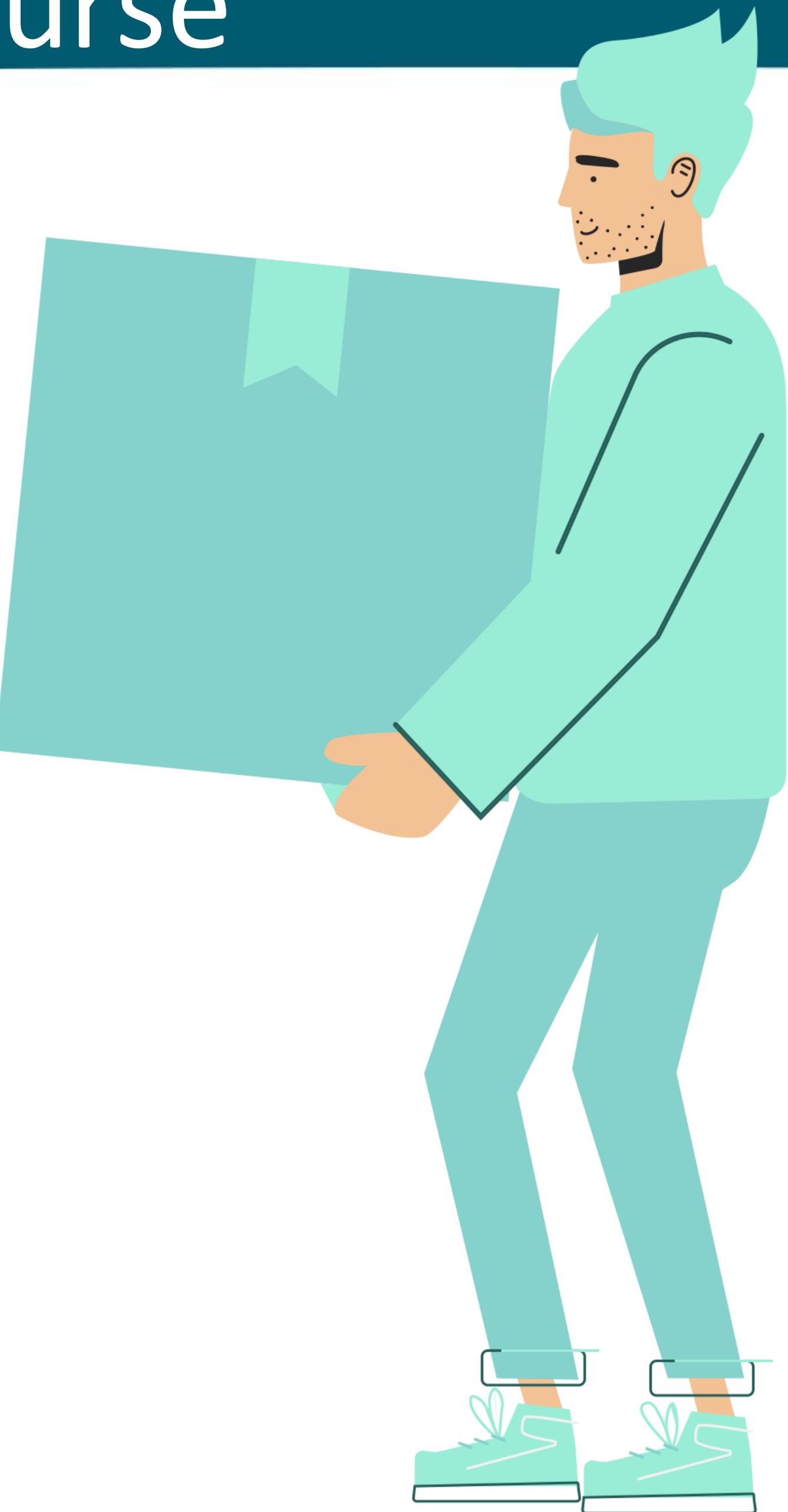
- **Roots**
- **Sampling**



# Key Takeaways from This Course

## PART 3

You'll learn hands-on how to build  
an MCP Server in **Python**



# MCP Inspector - How to Add a New MCP Server

MCP Inspector v0.16.1

Transport Type: STDIO

Command: uv

Arguments: --directory /Users/whitebeard/

> Environment Variables

Server Entry Servers File

> Configuration

Restart Disconnect

Connected

Error output from MCP server

Starting Weather MCP Server...  
Server ready to accept  
connections via STDIO

Tools

List Tools

Clear

Select a tool

Select a tool from the list to view its details and run it

History

1. initialize

Server Notifications

No notifications yet



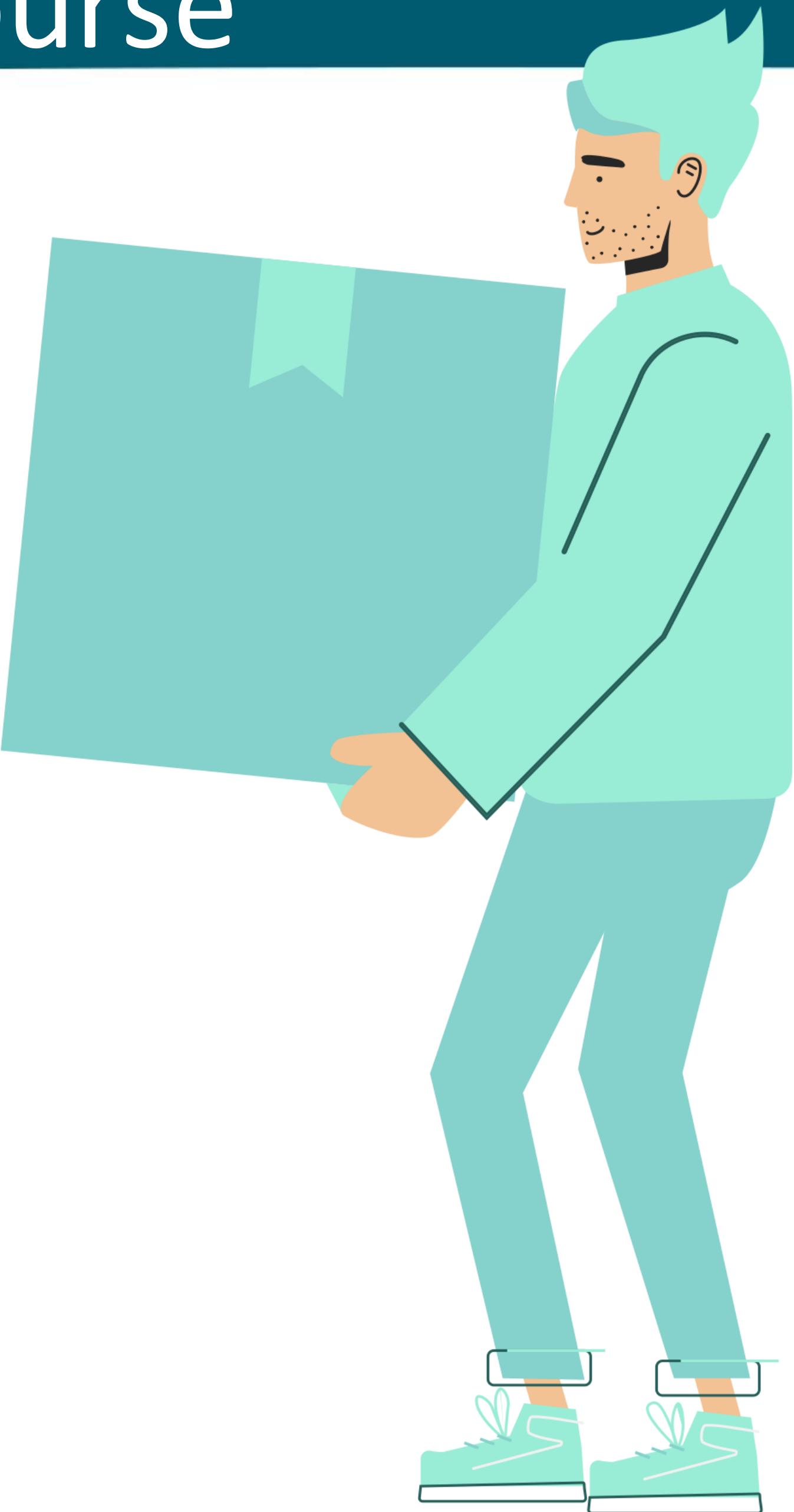
## MCP Inspector



# Key Takeaways from This Course

## PART 4

You'll learn hands-on how to build  
an MCP Server in **Java**



# Postman - How to Add a New MCP Server

The screenshot shows the Postman application window. In the top navigation bar, there are links for Home, Workspaces (with a dropdown menu), and API Network. A search bar is located at the top right. On the left side, there's a sidebar with tabs for Collections, Environments, Flows, and History. The main content area is titled "MCP 1". It displays a collection named "MCP Collection / Java STUDIO server". Under this collection, there is a single item named "Java STUDIO server". The "Message" tab is selected, showing a single request labeled "1. get\_weather". The description for this request is: "Get current weather information for a specified city. Returns temperature data for the requested location." Below the message tab, there are sections for "Response" and "Notifications". At the bottom of the main content area, there's a button labeled "Run" and a small icon of a character holding a sword. The footer of the Postman window contains various icons and links for Postbot, Runner, Start Proxy, Cookies, Vault, Trash, and Help.



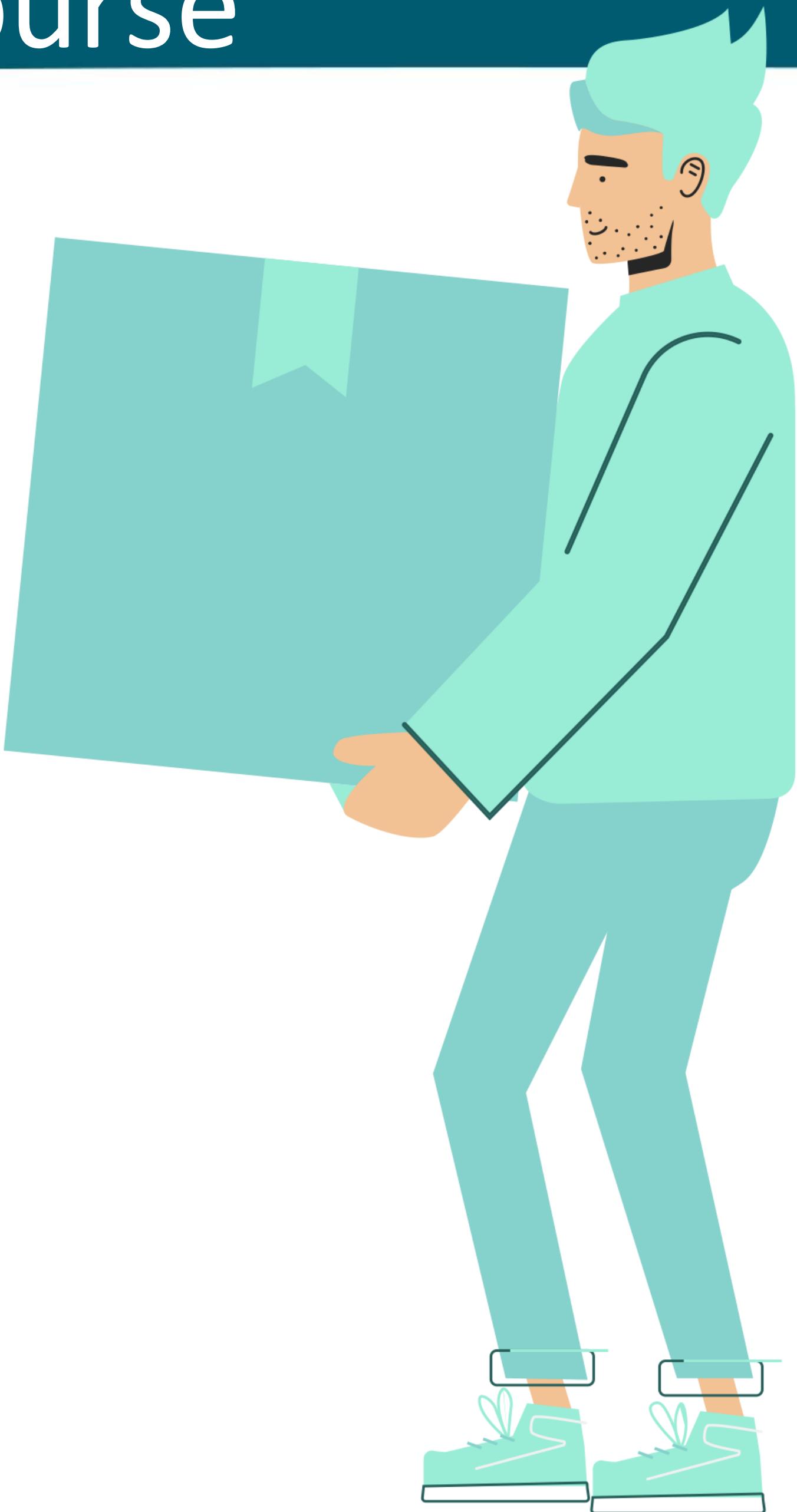
Postman



# Key Takeaways from This Course

## PART 5

You'll learn hands-on how to build  
an MCP Server in **Node.js**



# Claude - How to Add a New MCP Server

Dallas Weather Forecast ▾

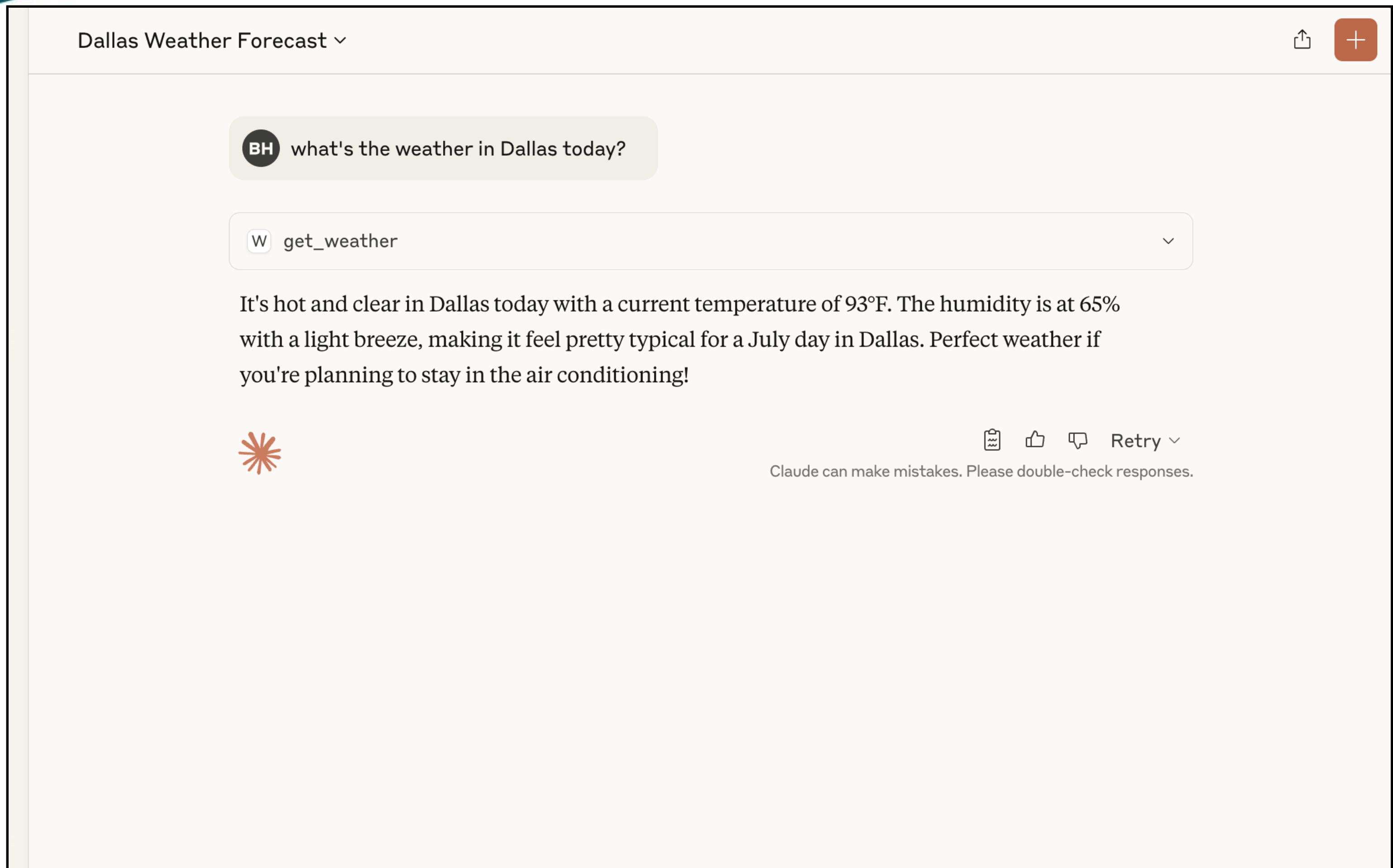
BH what's the weather in Dallas today?

W get\_weather

It's hot and clear in Dallas today with a current temperature of 93°F. The humidity is at 65% with a light breeze, making it feel pretty typical for a July day in Dallas. Perfect weather if you're planning to stay in the air conditioning!

✖️

Clipboard icon, Like icon, Dislike icon, Retry dropdown, Claude can make mistakes. Please double-check responses.



Claude Desktop

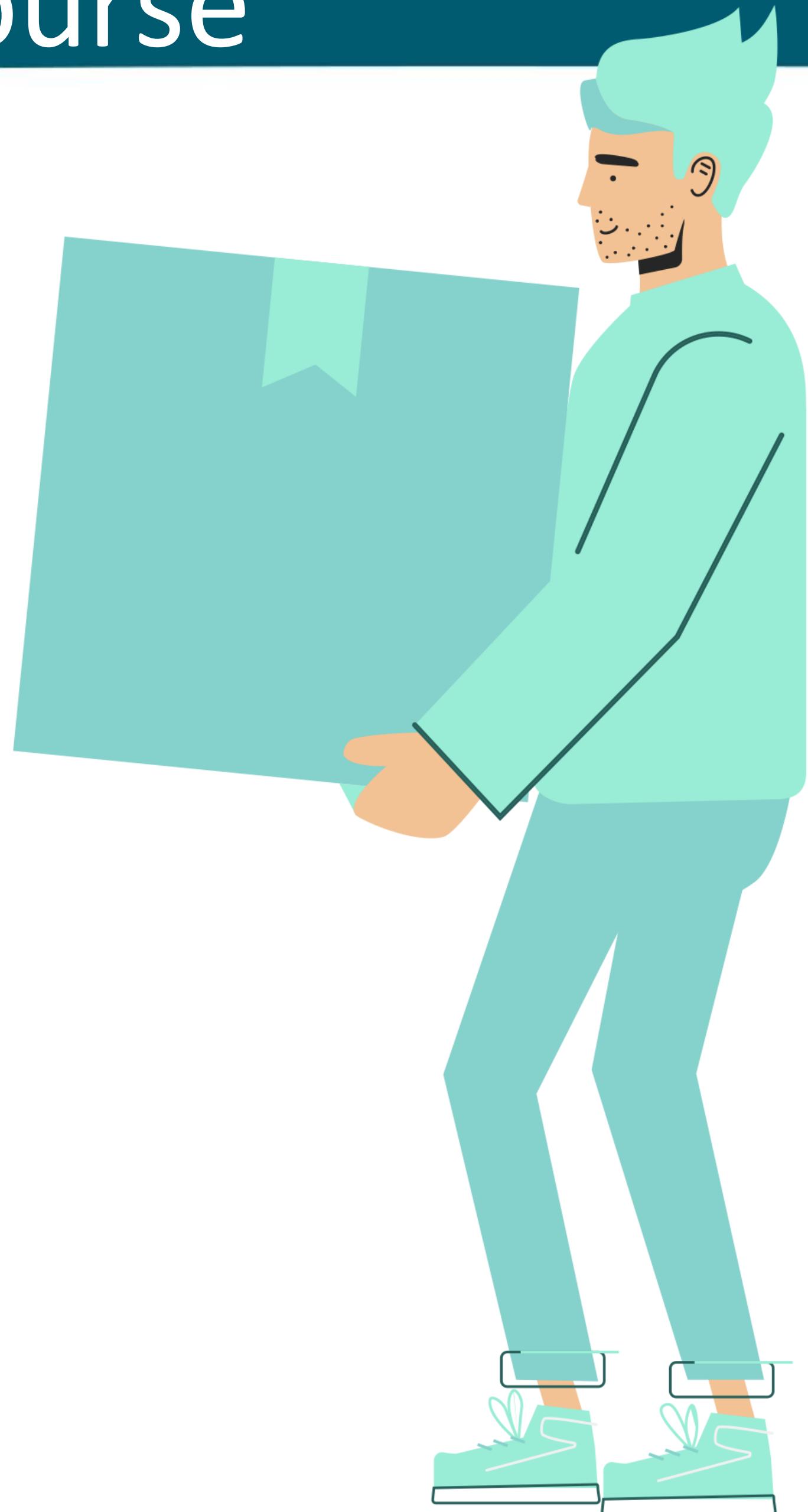


# Key Takeaways from This Course

## PART 6

We will switch focus from building MCP Servers from scratch to **leveraging existing** MCP Servers

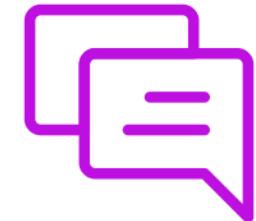
You will learn how to use and configure the **MCP Server for PostgreSQL**



# Ramifications



# DEMO 6 - Prompts Used



## **PROMPT:**

1. *"How many customers prefer online shopping vs retail stores?"*
2. *"What's the total revenue from online sales compared to retail sales?"*
3. *"Which products are most popular in retail stores?"*
4. *"Show me customers who prefer online shopping but have made purchases in retail stores"*
5. *"What's the average order value for each purchase location?"*
6. *"Which Nike products have generated the most revenue?"*
7. *"How many orders were placed in the last 30 days?"*
8. *"What's the distribution of customers across different states?"*
9. *"Which retail store has the highest sales volume?"*
10. *"Show me the top 5 customers by total spending"*

# Ramifications



- This will lead to a new generation of applications: NO User Interfaces

# Ramifications



- This will lead to a new generation of applications: NO User Interfaces
- The biggest impact: Internal corporate apps

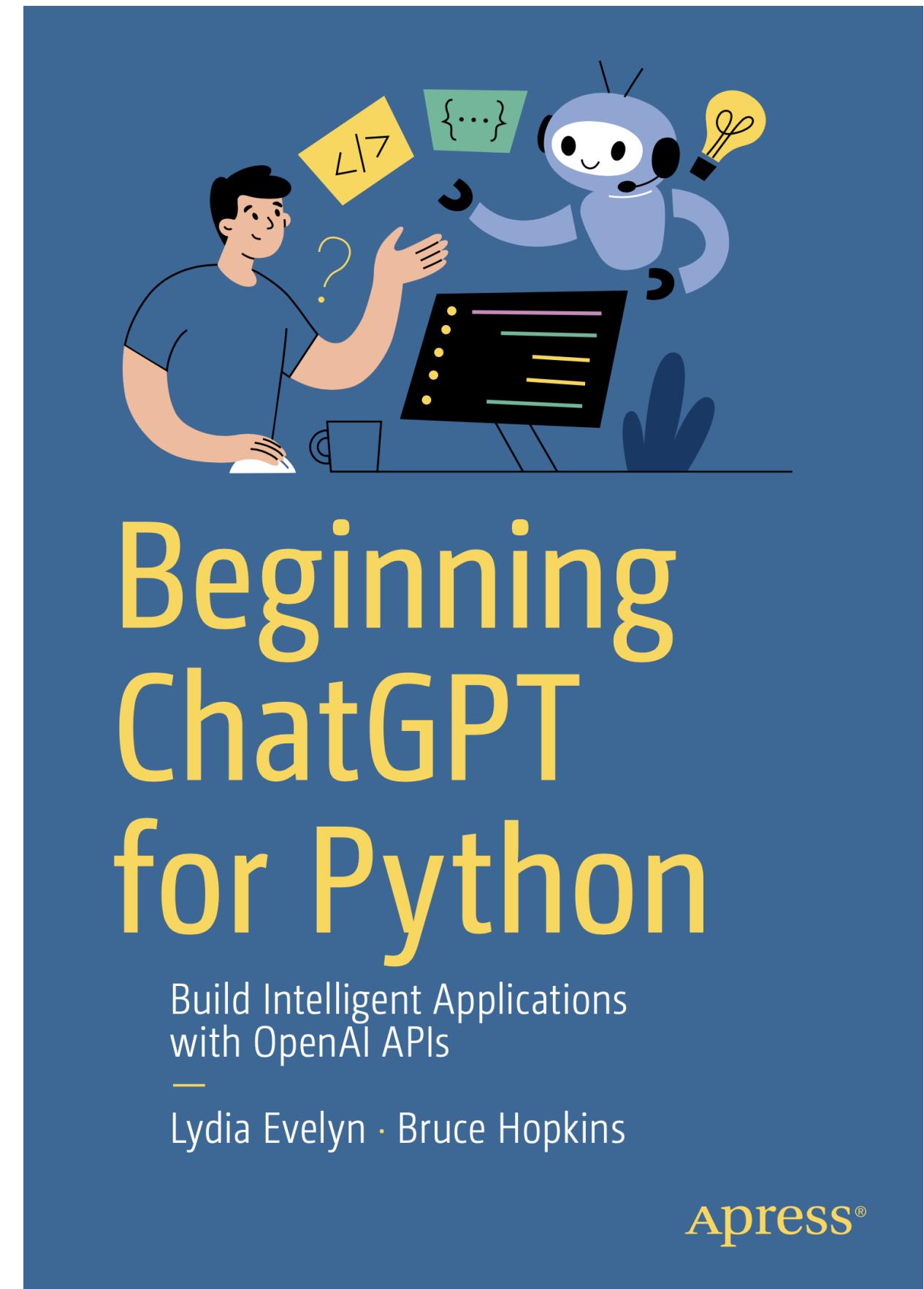
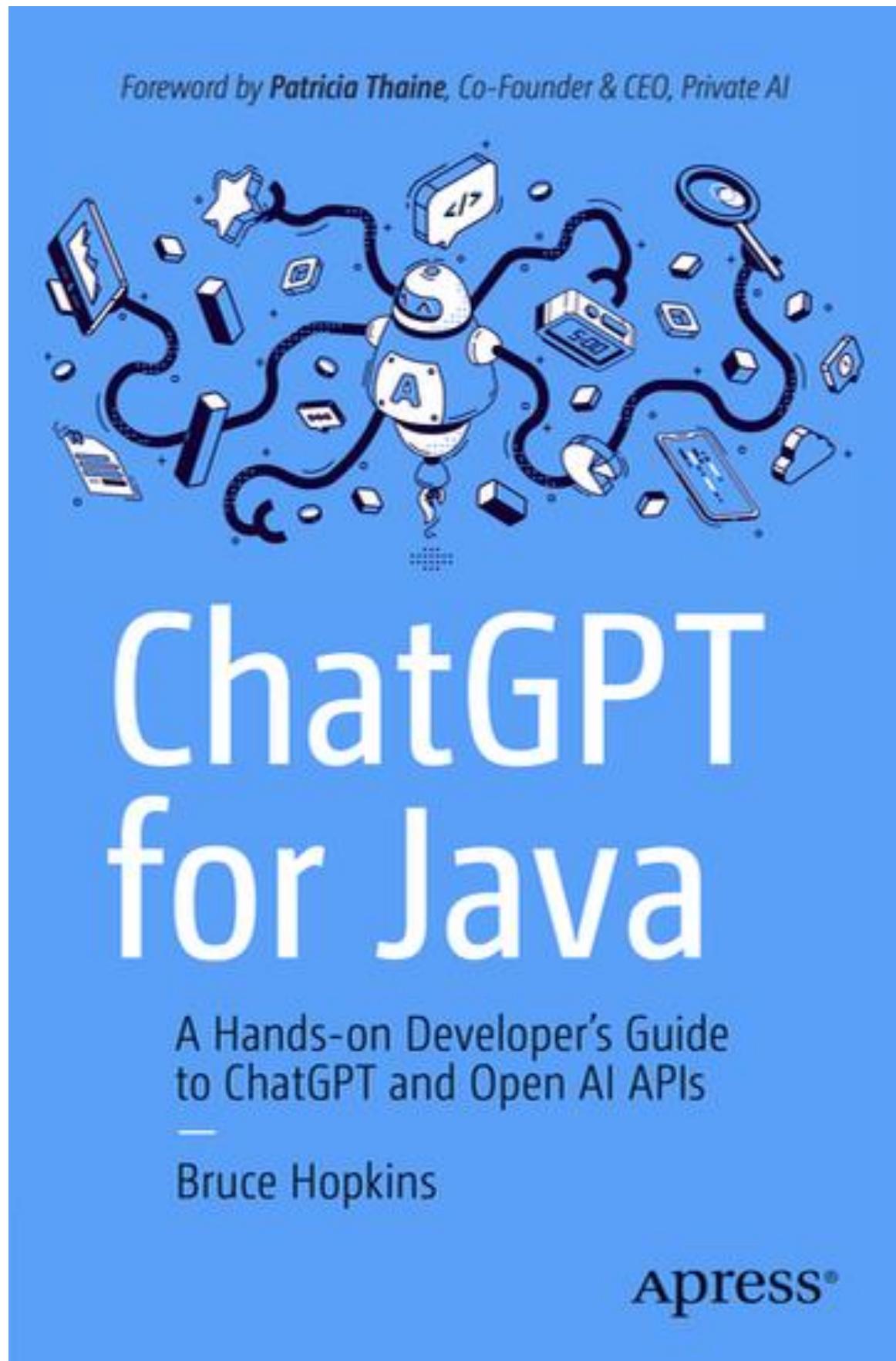
# Ramifications



- This will lead to a new generation of applications: NO User Interfaces
- The biggest impact: Internal corporate apps
- The demo used PostgreSQL, but the technique can be applied to **any database**

# Contact Info

- Twitter/ X  
[@JavaChatGPT](https://twitter.com/JavaChatGPT)
- Bluesky  
[@javachatgpt.bsky.social](https://bluesky.social/@javachatgpt.bsky.social)
- Github  
[https://github.com/BruceTraining/  
MCP-Oreilly-1](https://github.com/BruceTraining/MCP-Oreilly-1)



# So Much More to Cover

- Upcoming Training on **Oct 21**

The screenshot shows a course page for 'Create AI Agents with Model Context Protocol (MCP)'. At the top, there's a 'VIEW ALL EVENTS' link. The course title is 'Create AI Agents with Model Context Protocol (MCP)'. It's published by Pearson and is intermediate level. The description says 'Hands-on intro to AI Agents using the MCP standard'. Below this, there's a section titled 'Complete this course and earn a badge!' with a red circular icon. Underneath, there are links for 'What you'll learn', 'Is this live event for you?', and 'Schedule'. A bulleted list of learning objectives follows: 'Learn about the benefits and practical applications of AI Agents that leverage the Model Context Protocol (MCP)', 'Learn how to build and deploy MCP clients and servers.', and 'Get hands-on experience with creating AI Agents using the HTTP REST APIs in Python, Java 21, and Node.js.'. A note at the bottom states: 'Just as HTTP (Hyper Text Transfer Protocol) is the universal standard to connect web clients and servers, and FTP (File Transfer Protocol) is the universal standard to connect file client and servers, MCP (Model Context Protocol) is the emerging standard to connect AI Agent clients to AI Agent servers.' To the right of the course details, there's information about the event: 'July 29', '4pm-8pm Central European Summer Time', '136 Spots Remaining', and a 'Sign up!' button. Below this, there's a section for 'Your Instructor' featuring a profile picture of Bruce Hopkins and his bio: 'Bruce Hopkins is a technical writer and AI expert. He is an Intel Software Innovator for AI, as well as an Oracle Java Champion. Bruce is also the author of the books, *ChatGPT for Java and Bluetooth for Java*.'. There are also sections for 'Associated roles' (Android developer, Backend developer, Business analyst, Business intelligence analyst) and '+27 more'. At the bottom, it says 'Skill covered'.

- Transports (HTTP)
- Resources

# So Much More to Cover



MCP for the Smart Home



MCP for the Smart Car

# So Much More to Cover

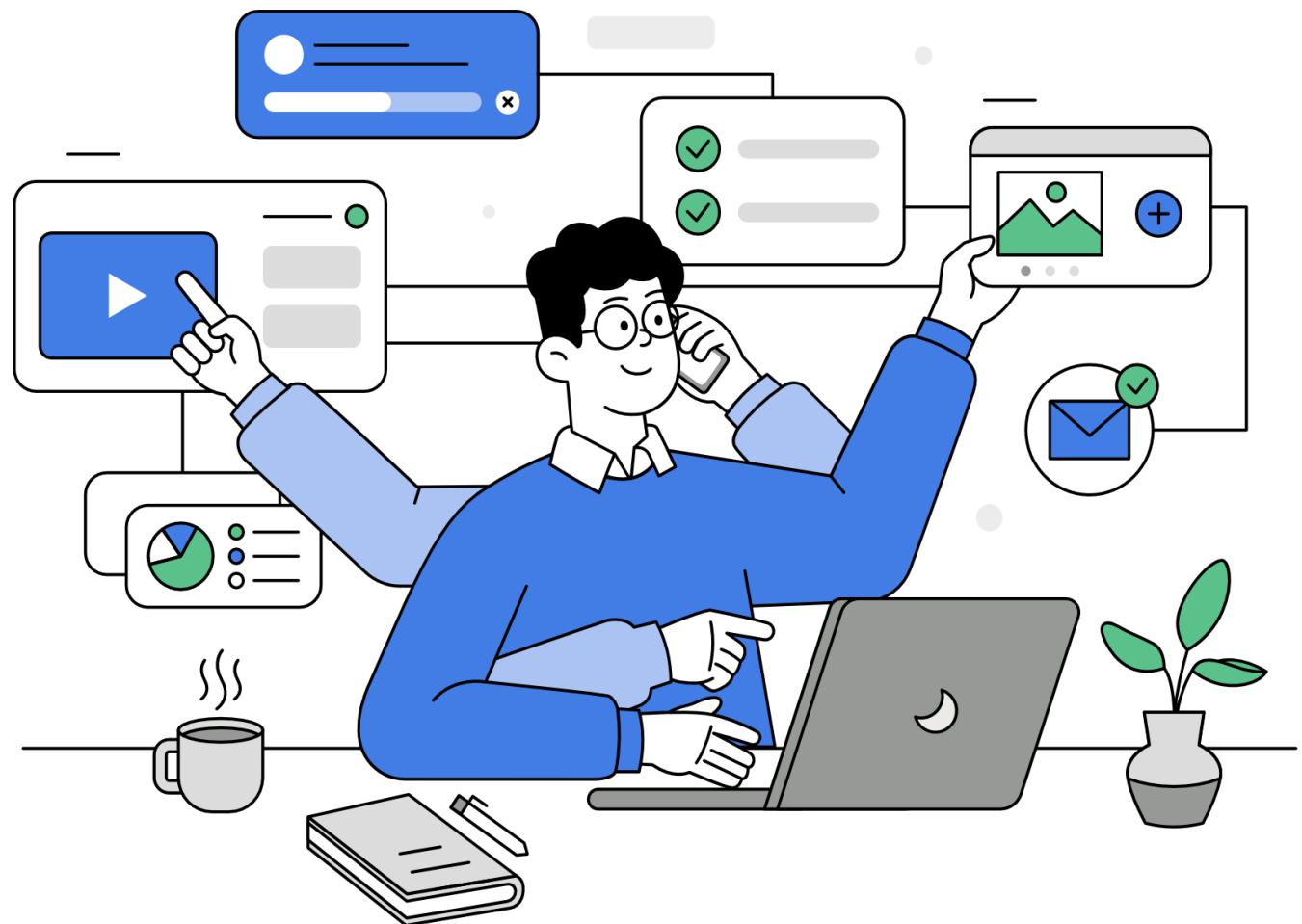


# So Much More to Cover

Coming Soon!

Dec 2025

THE ONLY BOOK YOU'LL NEED ON...



Creating **AI Agents** with

# MCP

JAVA 21

PYTHON

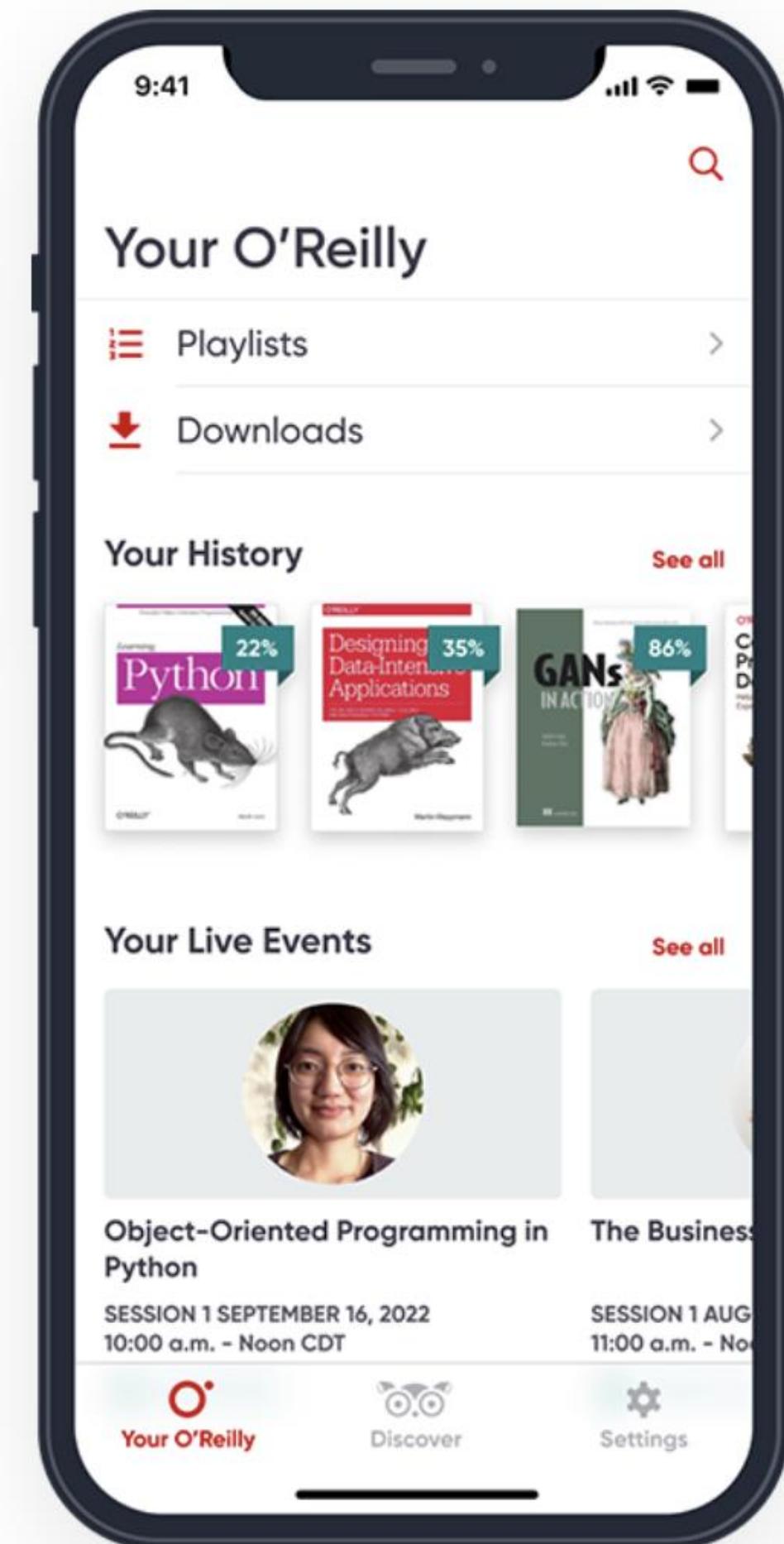
NODE.JS

**Lydia Evelyn · Bruce Hopkins**

# Have you downloaded the O'Reilly App?

## Mobile apps to learn on the go

Get unlimited access to books, audiobooks, courses, and live events on the go. You can download content for offline viewing. And devices automatically sync, so you'll never lose your place. Plus, you can create and organize collections of your favorite resources with playlists.



Do me a favor...

O'Reilly wants to hear from you!