DUTCH AZURE MEETUP

GitHub Copilot ++ with MCP for Azure

A presentation by Rob Bos & Hidde de Smet

Introduction

Rob Bos

DevOps Consultant | GitHub Trainer

Hidde de Smet

Azure Architect | Trainer

Powered by Xebia

Talking points:

What is GitHub Copilot ?

What is MCP?

Demo’s!

A look ahead

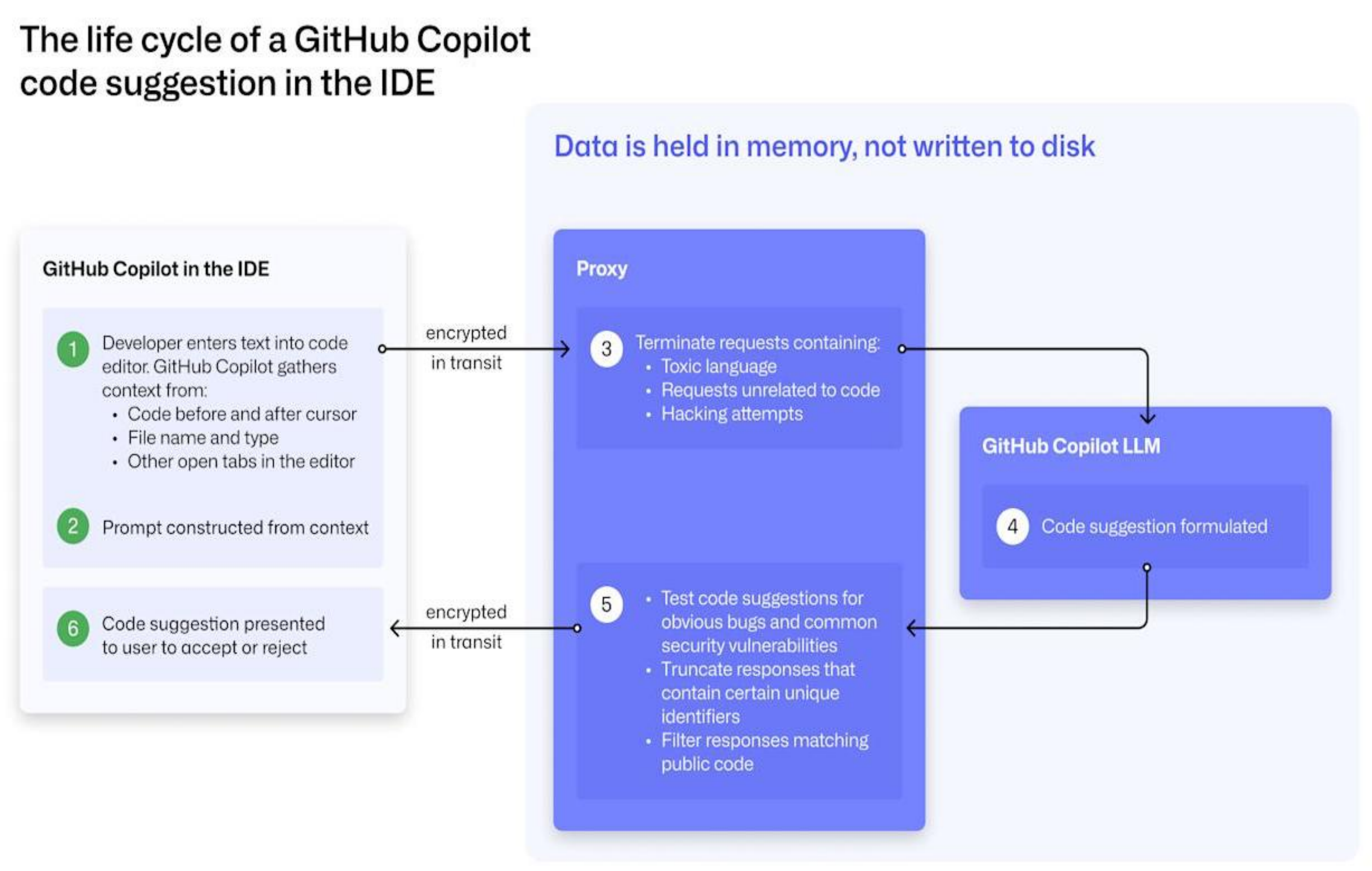
What is GitHub Copilot?

What is GitHub Copilot?

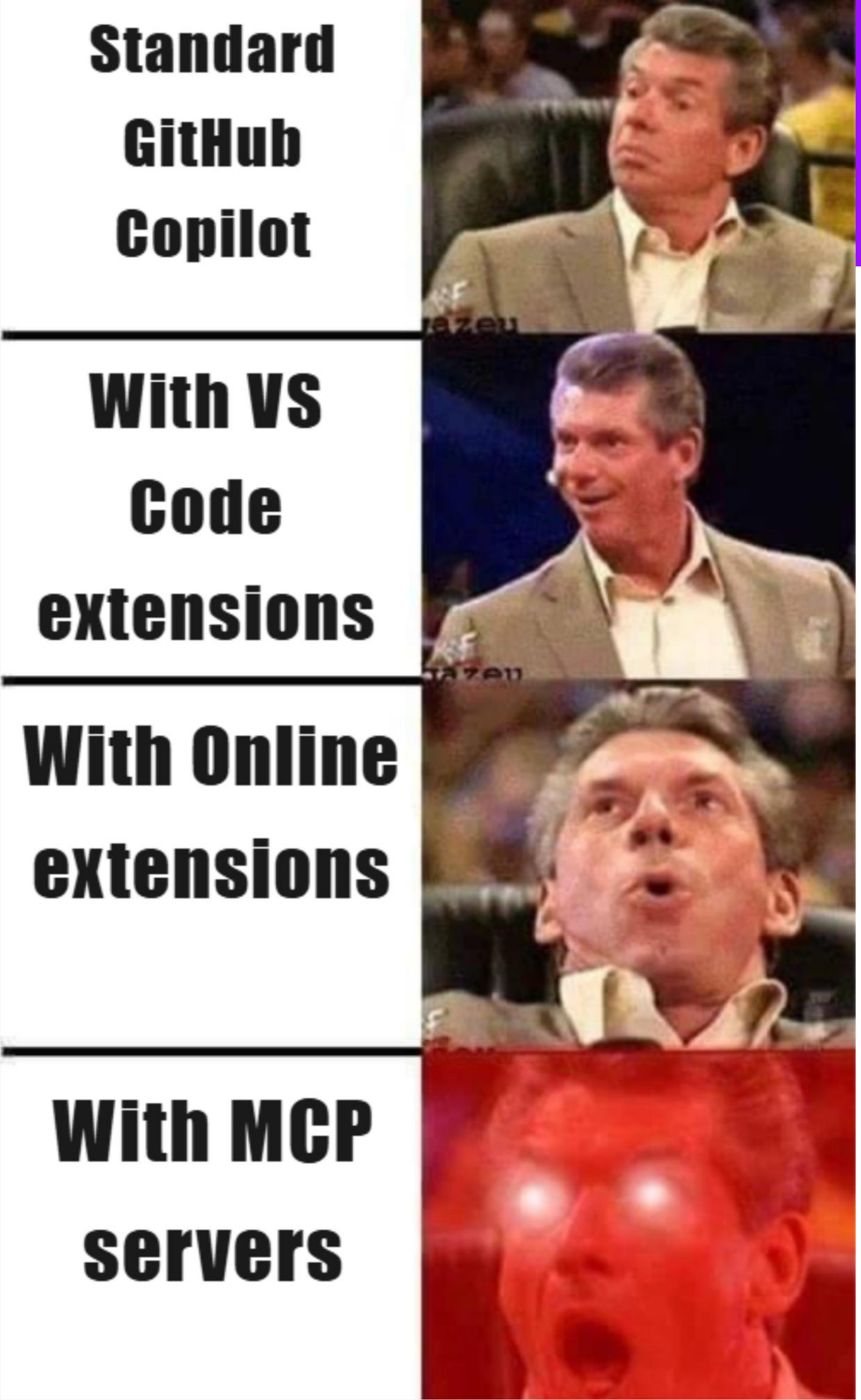
"An AI-powered code completion tool that helps developers write code faster."

"AI powered Pair Programmer"

How does GitHub Copilot work?



The life cycle of a GitHub Copilot code suggestion in the IDE
Data is held in memory, not written to disk
GitHub Copilot in the IDE
1 Developer enters text into code editor. GitHub Copilot gathers context from:
• Code before and after cursor
• File name and type
• Other open tabs in the editor
2 Prompt constructed from context
6 Code suggestion presented to user to accept or reject
encrypted in transit
encrypted in transit
Proxy
3 Terminate requests containing:
• Toxic language
• Requests unrelated to code
• Hacking attempts
5
• Test code suggestions for obvious bugs and common security vulnerabilities
• Truncate responses that contain certain unique identifiers
• Filter responses matching public code
GitHub Copilot LLM
4 Code suggestion formulated



Why use MCP?

MCP helps you build agents and complex workflows on top of LLMs.

LLMs frequently need to integrate with data and tools, and MCP provides:

- A growing list of pre-built integrations that your LLM can directly plug into
- The flexibility to switch between LLM providers and vendors
- Best practices for securing your data within your infrastructure

What is MCP?

Model Context Protocol

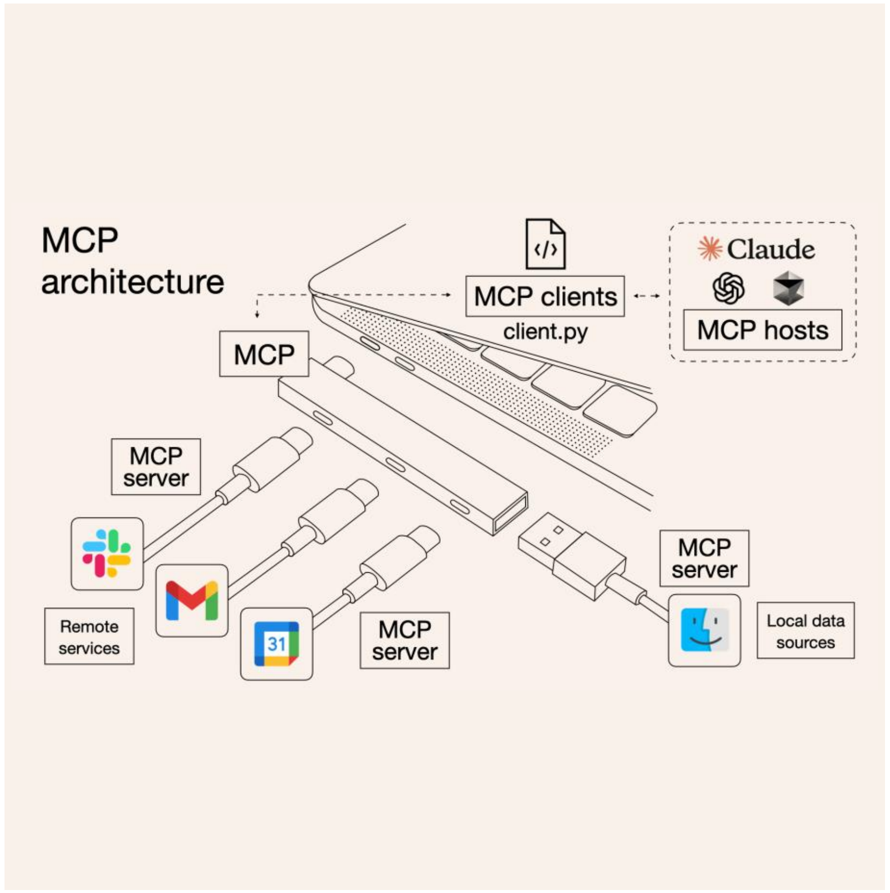
Demo time!

What is MCP? Demo time!

Topics:

- Add more GitHub Repo context
- Load Azure Resources context

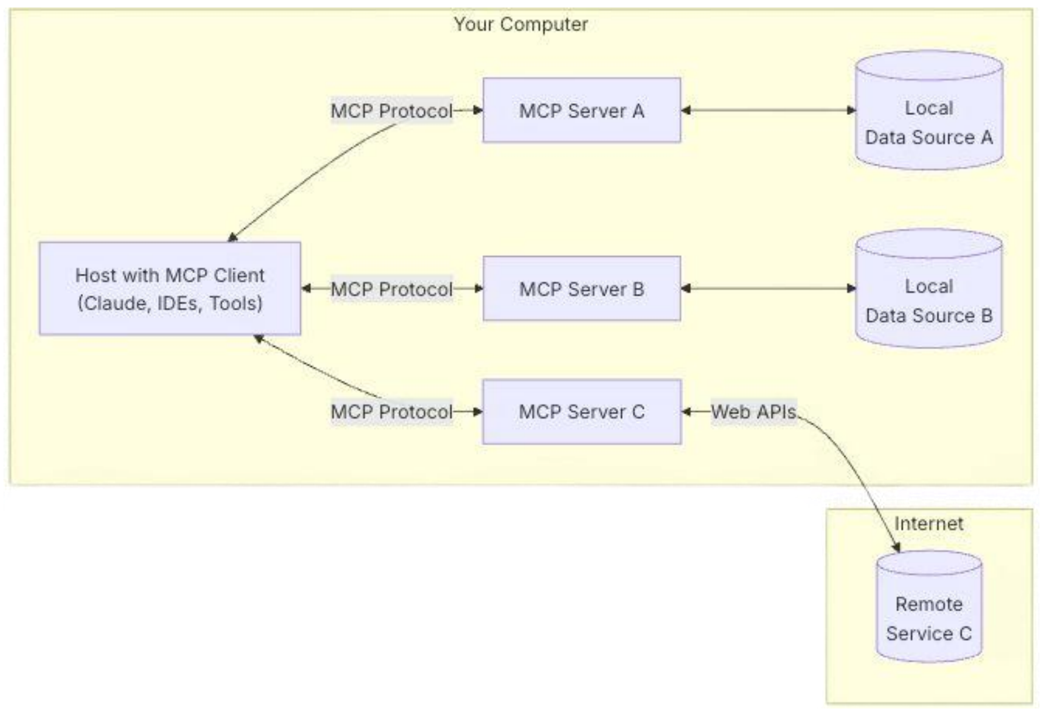
What is MCP?



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

2. The protocol can also be leveraged to add **execution** options to the clients.

MCP Architecture



At its core, MCP follows a client-server architecture where a host application can connect to multiple servers:

- **MCP Hosts:** Programs like Claude Desktop, IDEs, or AI tools that want to access data through MCP
- **MCP Clients:** Protocol clients that maintain 1:1 connections with servers
- **MCP Servers:** Lightweight programs that each expose specific capabilities through the standardized Model Context Protocol
- **Local Data Sources:** Your computer's files, databases, and services that MCP servers can securely access
- **Remote Services:** External systems available over the internet (e.g., through APIs) that MCP servers can connect to

Host

- Creates and manages multiple client instances
- Controls client connection permissions and lifecycle
- Enforces security policies and consent requirements
- Handles user authorization decisions
- Coordinates AI/LLM integration and sampling
- Manages context aggregation across clients

Clients

- Establishes one stateful session per server
- Handles protocol negotiation and capability exchange
- Routes protocol messages bi-directionally
- Manages subscriptions and notifications
- Maintains security boundaries between servers

Servers

- Expose resources, tools and prompts via MCP primitives
- Operate independently with focused responsibilities
- Request sampling through client interfaces
- Must respect security constraints
- Can be local processes or remote services

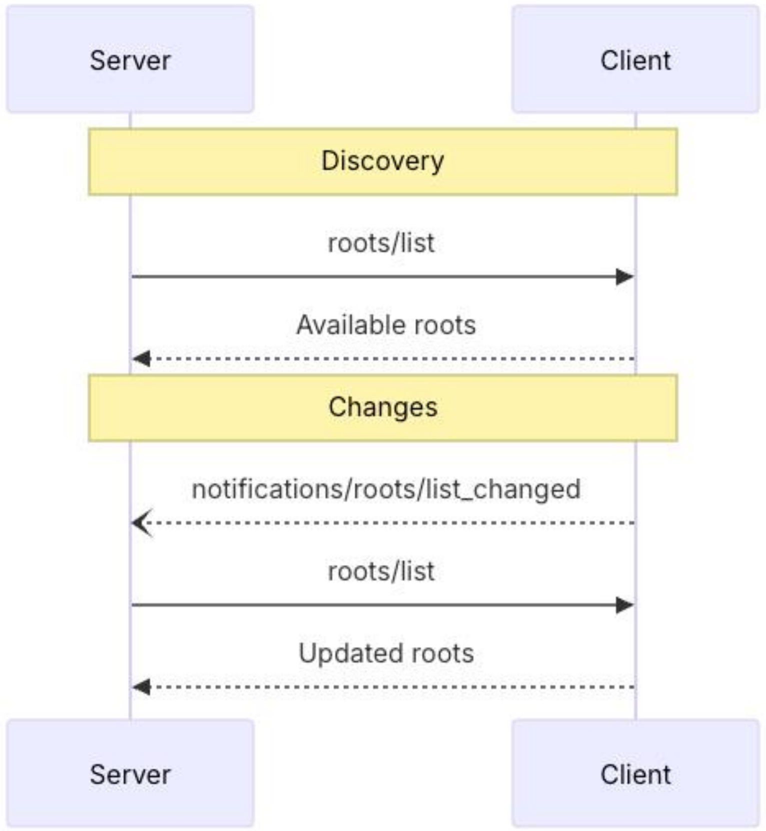
Client features overview

- **Roots**

  Roots define the boundaries of where servers can operate within the filesystem, allowing them to understand which directories and files they have access to.

- **Sampling**

  The Model Context Protocol (MCP) enables clients to manage model access, selection, and permissions, allowing servers to leverage AI capabilities without API keys. Servers can request text, audio, or image interactions with optional context from MCP servers.



Server features overview

Primitive	Control	Description	Example
Prompts	User-controlled	Interactive templates templates invoked by by user choice	Slash commands, menu options
Resources	Application-controlled	Contextual data attached and managed by the client	File contents, git history
Tools	Model-controlled	Functions exposed to the LLM to take actions	API POST requests, file writing

Why MCP isn't secure (yet)

- Easy integrations
- Unified interfaces
- No authentication standard
- No context encryption
- No way to verify tool integrity

THE "S" IN MCP STANDS FOR SECURITY
I CALL IT AN OPEN INVITATION TO CHAOS.
imgflip.com

Demo's

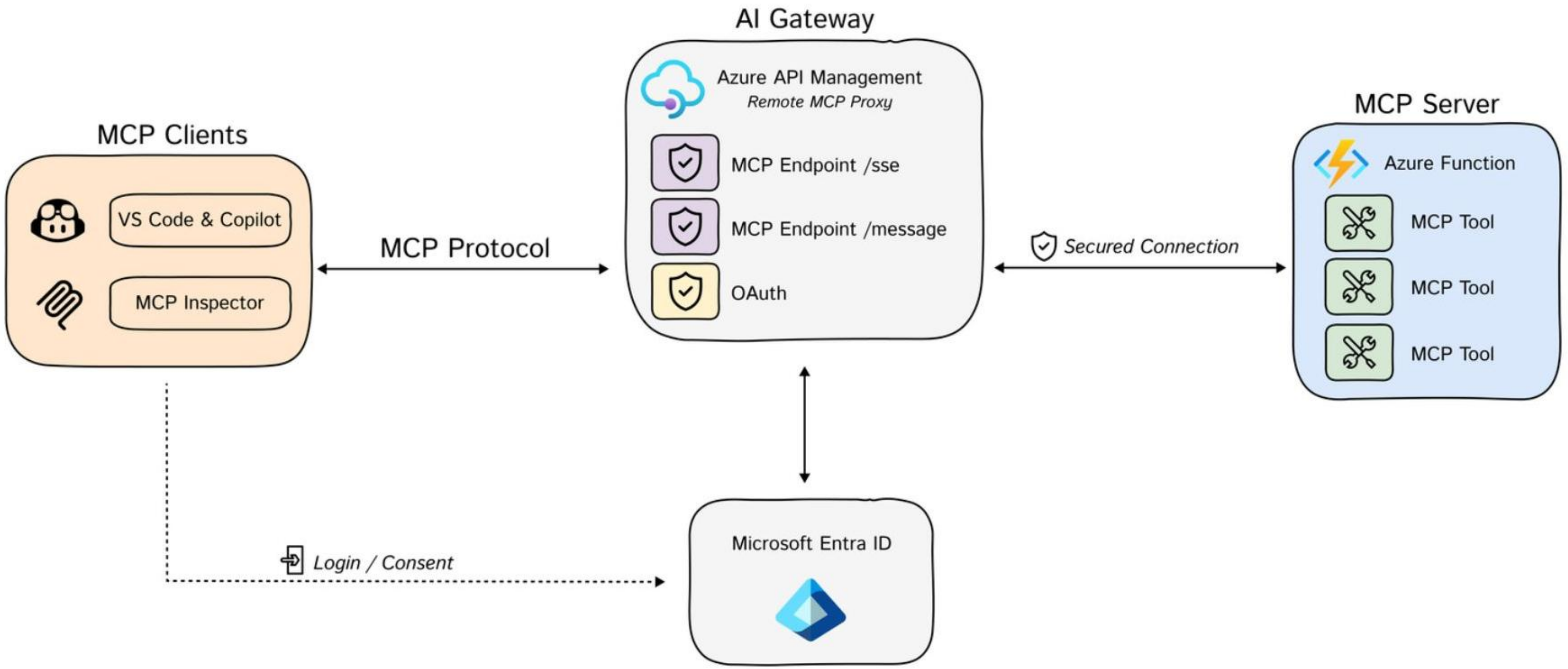
Demo's

<u>**End to end demo:**</u>

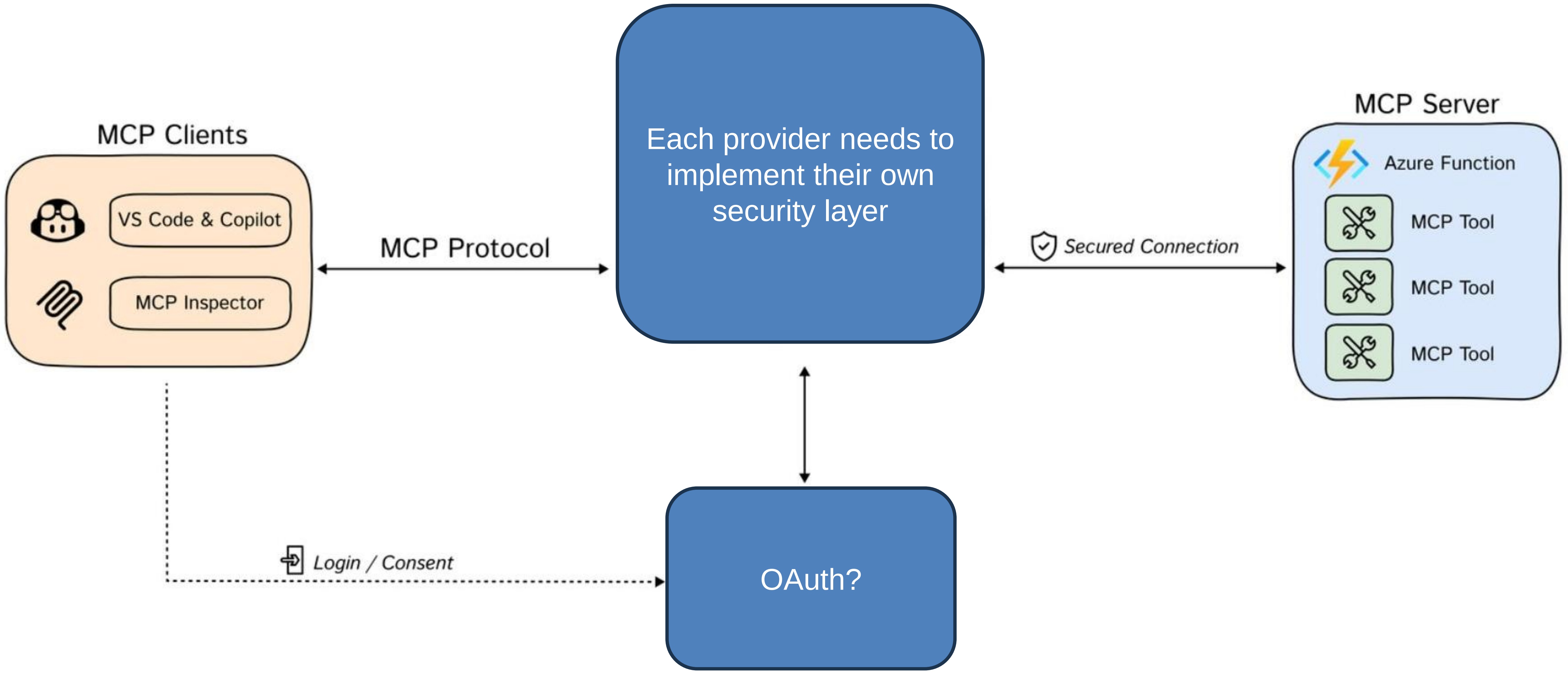
- Load error information from Application Insights
- Create issues for these errors
- Pick an issue
- Implement the change to fix the issue
    - Include creating a PR, link the PR to the issue

Looking ahead

Future scenarios



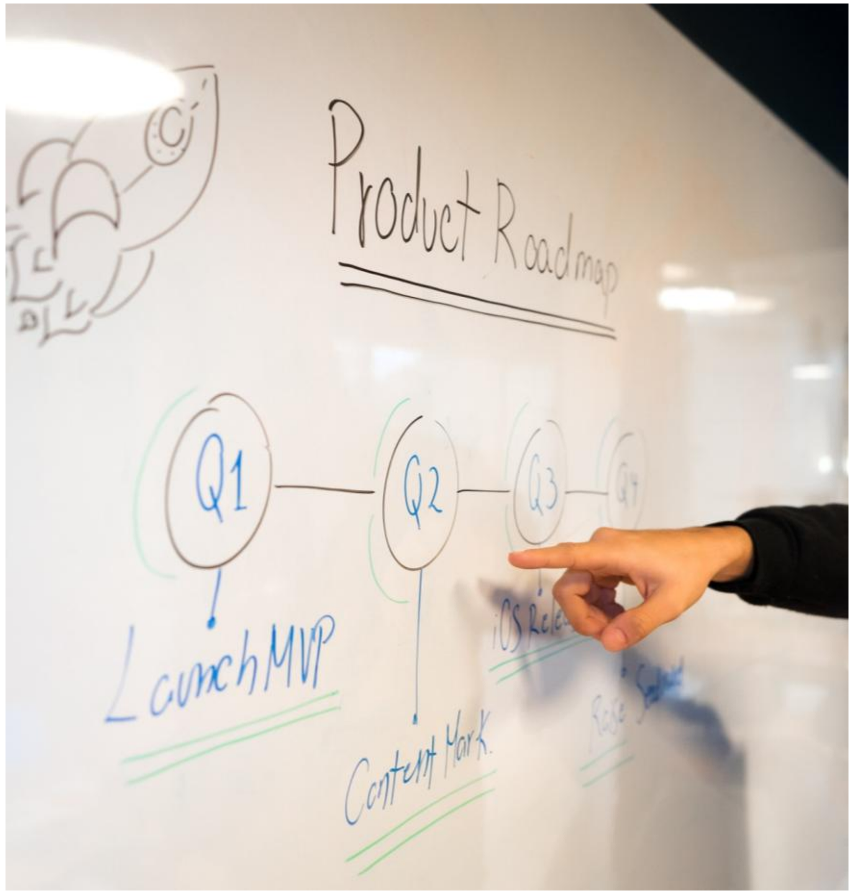
Future scenarios



Roadmap

- Validation
- Registry
- Agents – Computers Using Agents (CUA)
- Multimodality
- Governance
- Get involved

  https://github.com/orgs/modelcontextprotocol/discussions



Summary

How to get started with MCP

- https://modelcontextprotocol.io/introduction

  Explanation of what MCP is, how the architecture works etc.

- https://github.com/modelcontextprotocol/servers

  Great overview of MCP servers and links to their respective codebases in GitHub.

- https://mcpagents.dev

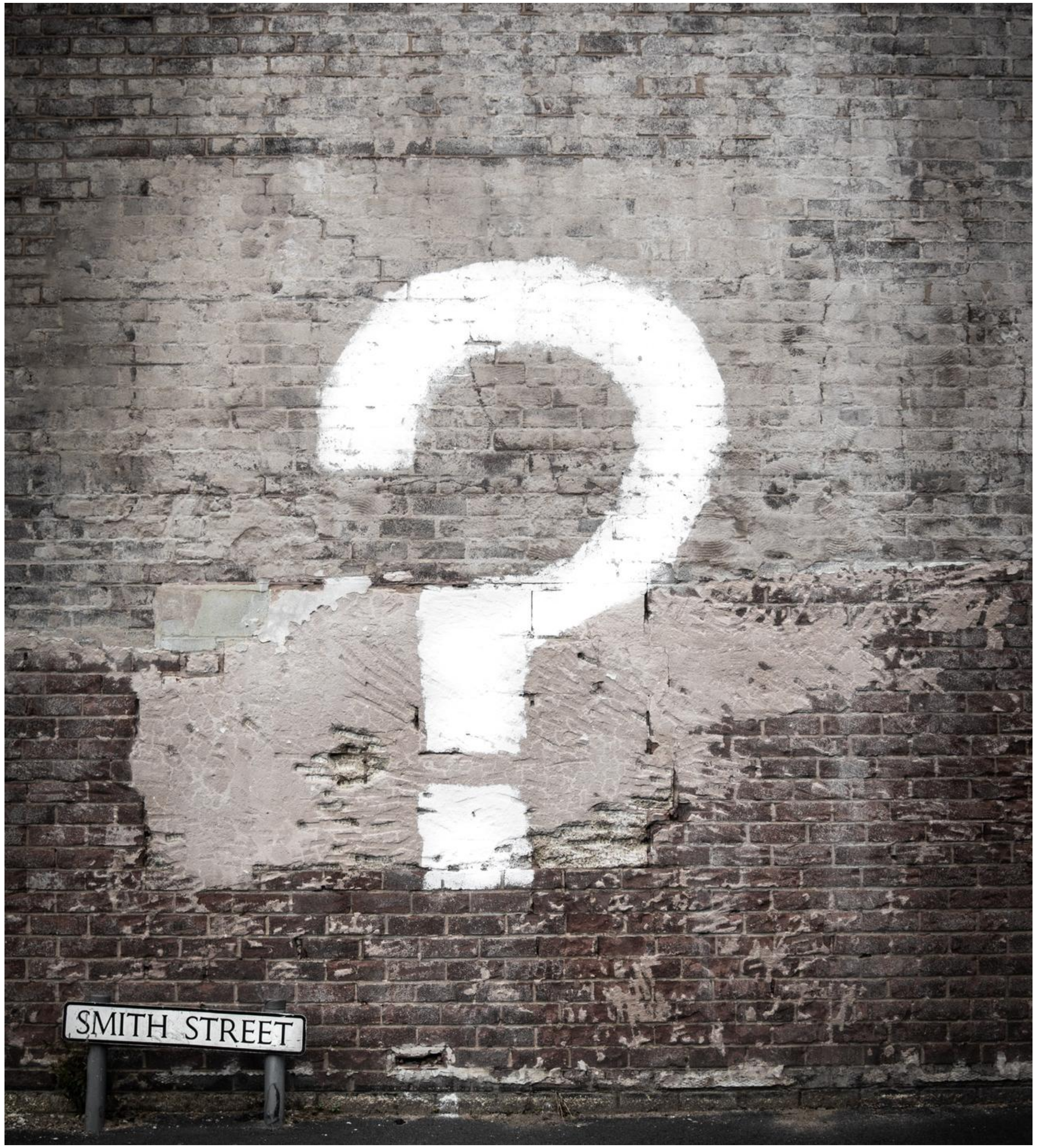
  Great overview of all kinds of MCP servers.

- https://glama.ai/mcp/servers

  Great overview of all kinds of MCP servers.

- https://www.youtube.com/watch?v=iS25RFups4A&t=914s

  Short tutorial to get you started.



Questions?

(Download the slides here)

QR Code!

GitHub Copilot and MCP for Azure

Rob Bos

DevOps Consultant | GitHub Trainer

https://devopsjournal.io

Powered by Xebia

Hidde de Smet

Azure Architect | Trainer