MCP in Action

Powering Single & Multi-Agent Systems with LangChain.js

Microsoft Reactor

Who are we?

Wassim Chegham
Senior Developer Advocate
Microsoft

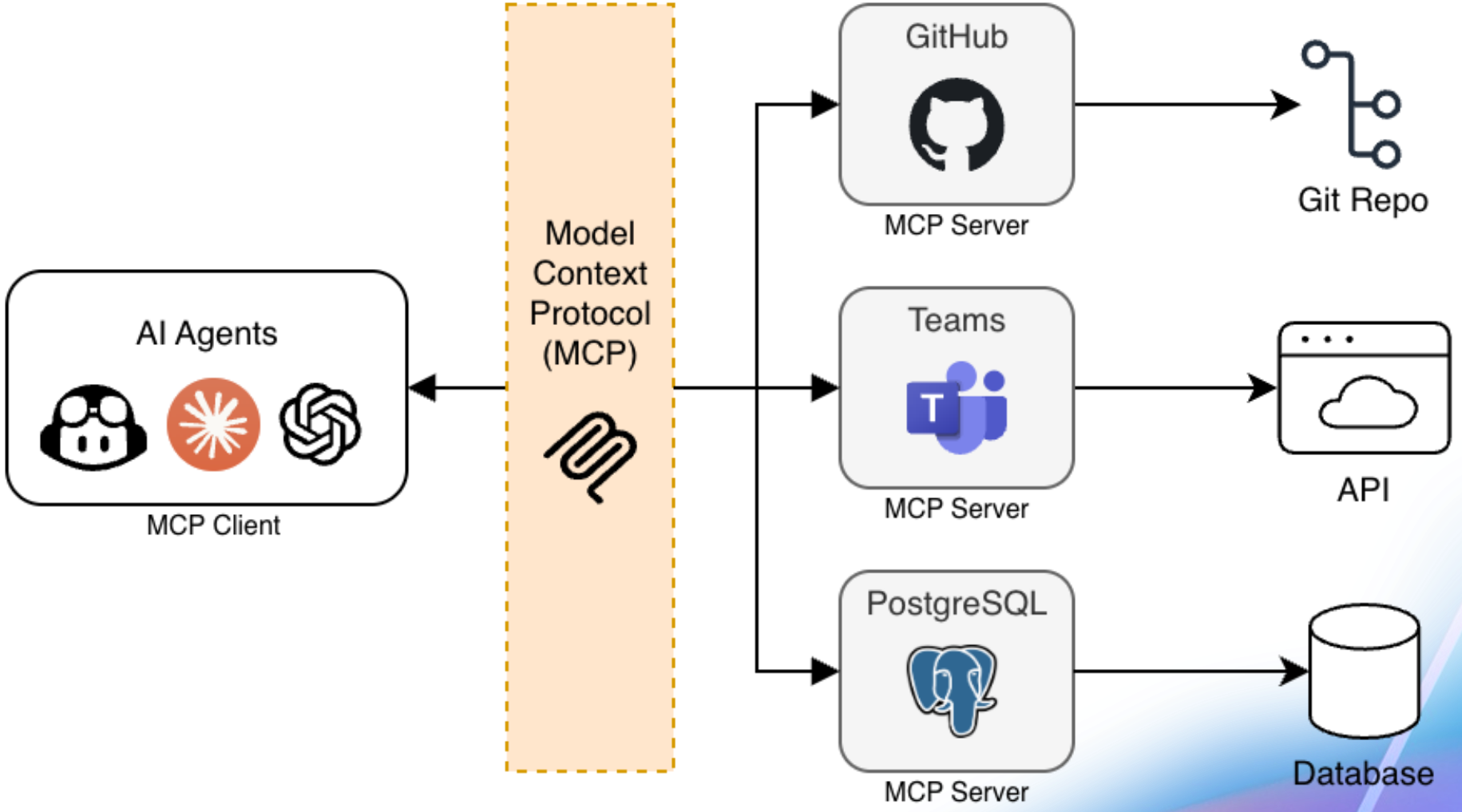
Yohan Lasorsa
Principal Developer Advocate
Microsoft

MCP in Action

(Short) Introduction to MCP

aka.ms/oss-ai-summit

What is MCP?



aka.ms/oss-ai-summit

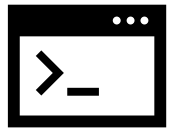
Using MCP

Client and server SDKs

- TypeScript
- Python
- Go
- Kotlin
- Swift
- Java
- C#
- Ruby
- Rust
- PHP

Transports

- stdio
  → *local servers*



- Streamable HTTP
  → *remote servers*

MCP Features

Server

- Tools
- Resources
- Prompts

Client

- Roots
- Elicitation
- Sampling

MCP Security

- Uses OAuth 2.1 Authorization Framework
- Secures access to sensitive resources and operations exposed by MCP servers.
- Read official docs for code samples

  https://modelcontextprotocol.io/docs/tutorials/security/authorization

Demo

MCP Server “Hello World”

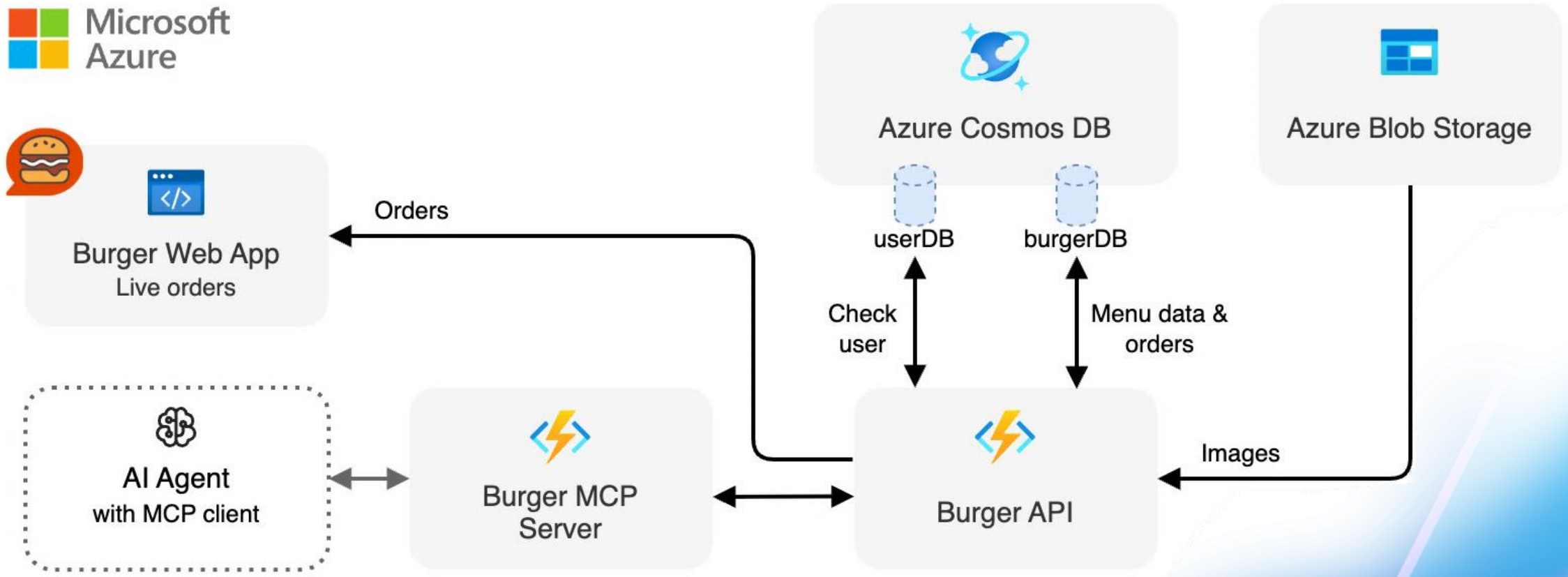
aka.ms/oss-ai-summit

Demo

Single Agent system with LangChain.js

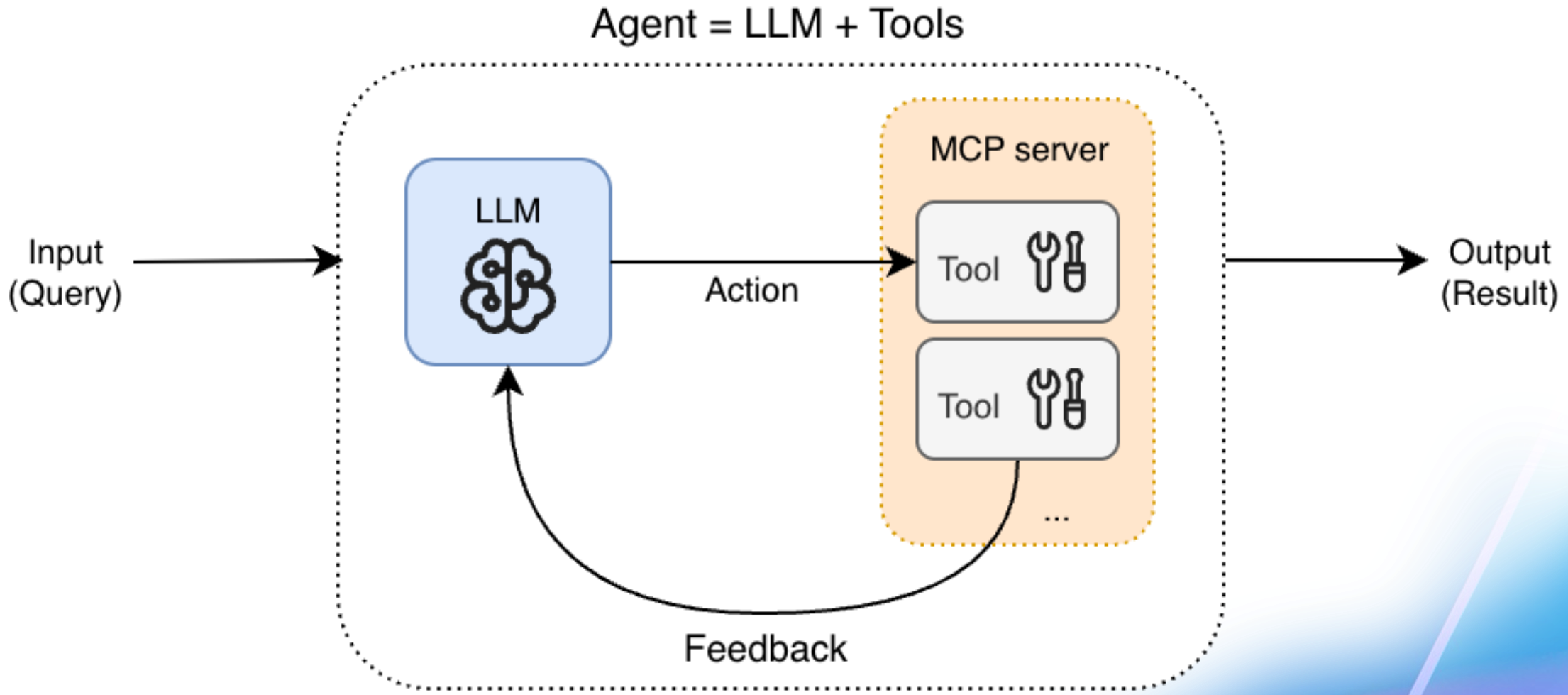
aka.ms/oss-ai-summit

High level architecture



aka.ms/oss-ai-summit

Simple agentic workflow



Overview: Burger Agent with MCP

Components: Burger API, MCP server, Agent API, Web Apps

Security: auth handled by Burger API

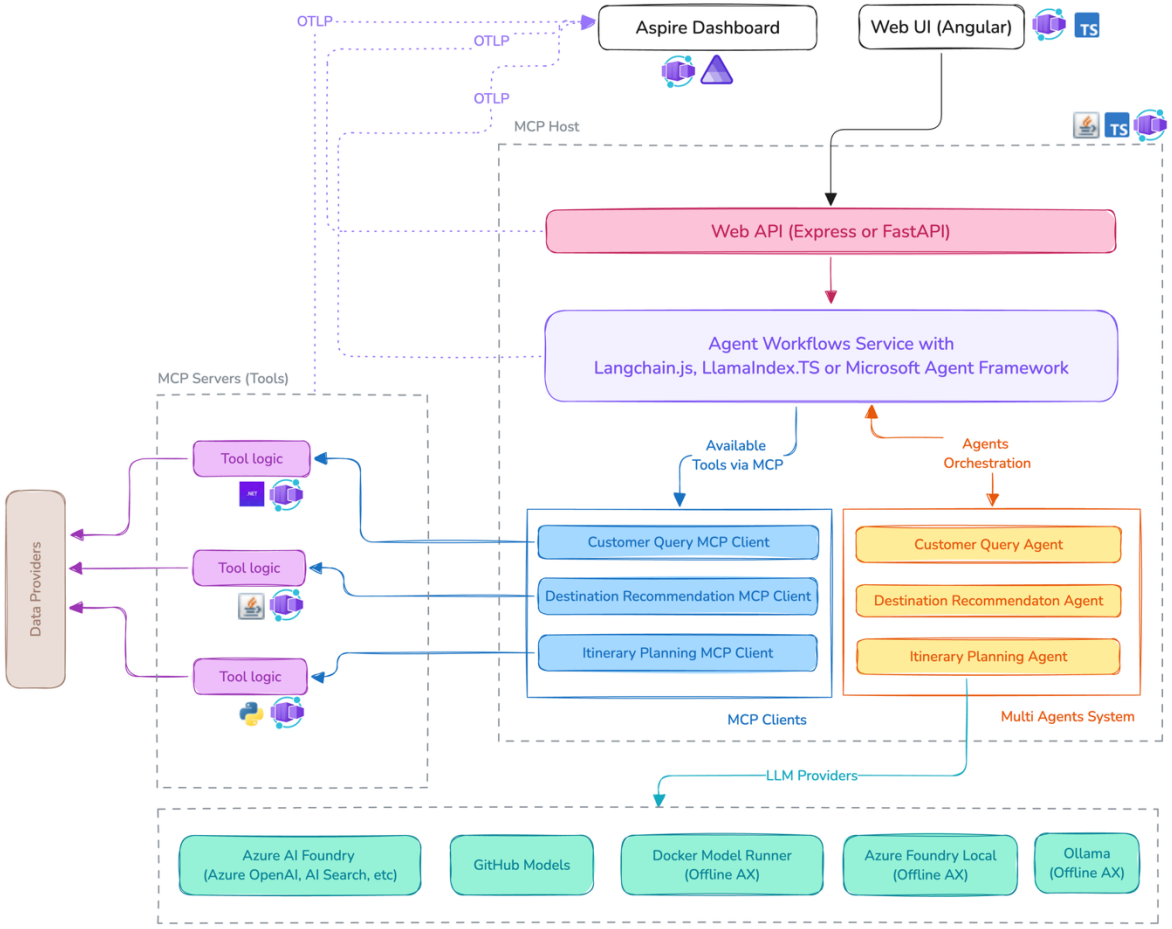
Scaling: stateless component + serverless hosting

Demo

Multi-Agent Orchestration with LangChain.js

aka.ms/oss-ai-summit

Real World Architecture Demo



aka.ms/oss-ai-summit

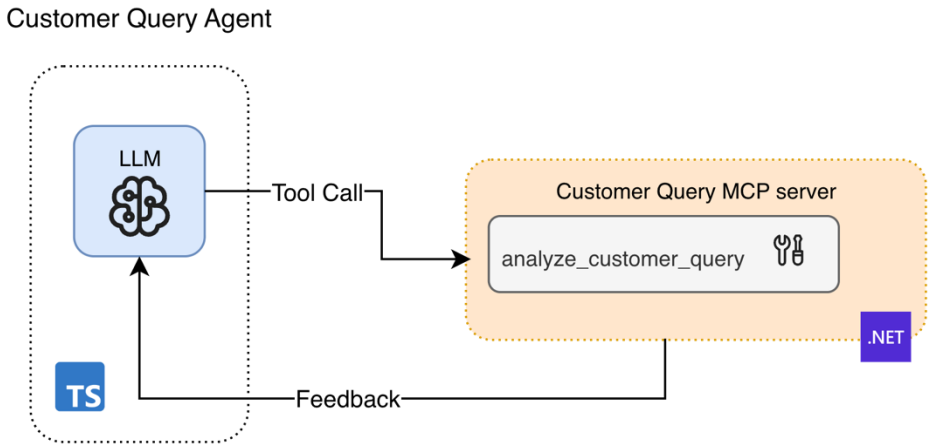
Why Multi-Agent systems?

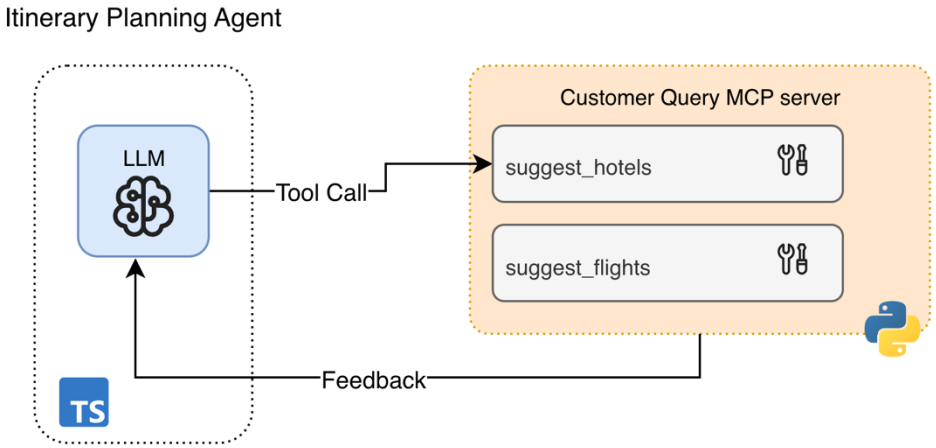
MAS is a system composed of multiple autonomous and specialized agents that interact together in shared environment to achieve a common objective.

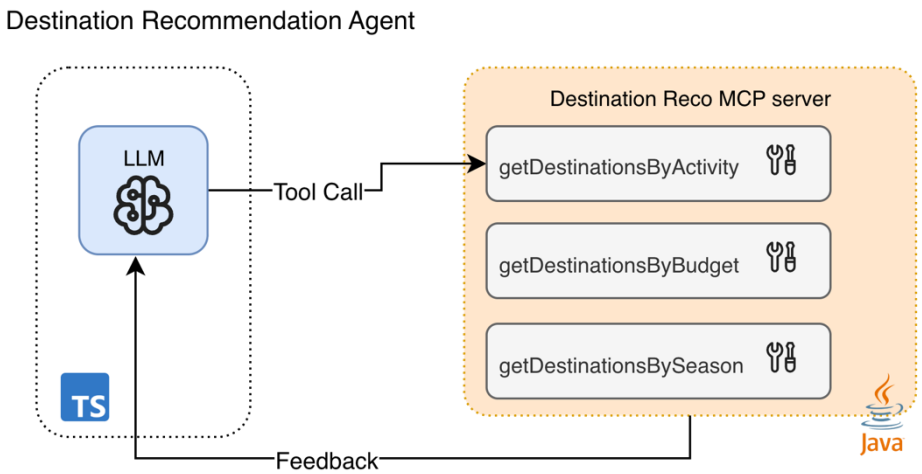
MAS are better suited for complex business process.

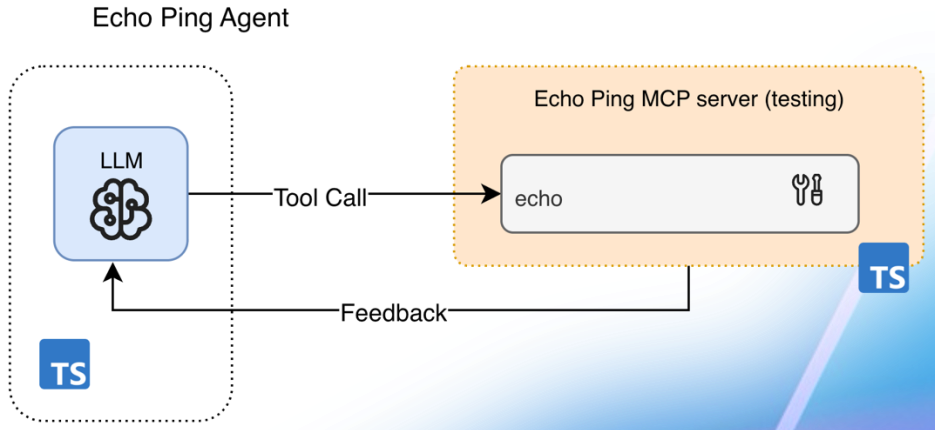
DeepAgents and *SubAgents* are great for controlling context.

MCP servers & multi-language support









aka.ms/oss-ai-summit

Security Considerations

- Level 0: Authentication with JWT (see demo)
- Level 1: Authorization
- Level 2: OAuth 2.1 – MCP recommendations

Recap’

Powering Single & Multi-Agent Systems with LangChain.js

aka.ms/oss-ai-summit

Recap': Agentic Systems with LangChain.js

MCP	• Allow agents to interact with your environment
Single Agent	• Simpler and more predictable, best for contained domains • Use `createAgent()`
Multi-Agent	• More performant for complex, dynamic domains • Use `deepagents` or `langgraph-supervisor`

Thank you!

Any question? Join us on Microsoft Foundry Discord

aka.ms/oss-ai-summit