ArchitectNow

Integrating Internal Systems with AI Agents

Real-World Applications of MCP and Microsoft Copilot

July 17, 2025

Presenter

Kevin Grossnicklaus
kvgros@architectnow.net
www.architectnow.net

Terminology

- Application Programming Interface (API)
- Model Context Protocol (MCP)
  - Anthropic
- Large Language Model (LLM)
- Agent (Agentic)
- Copilot

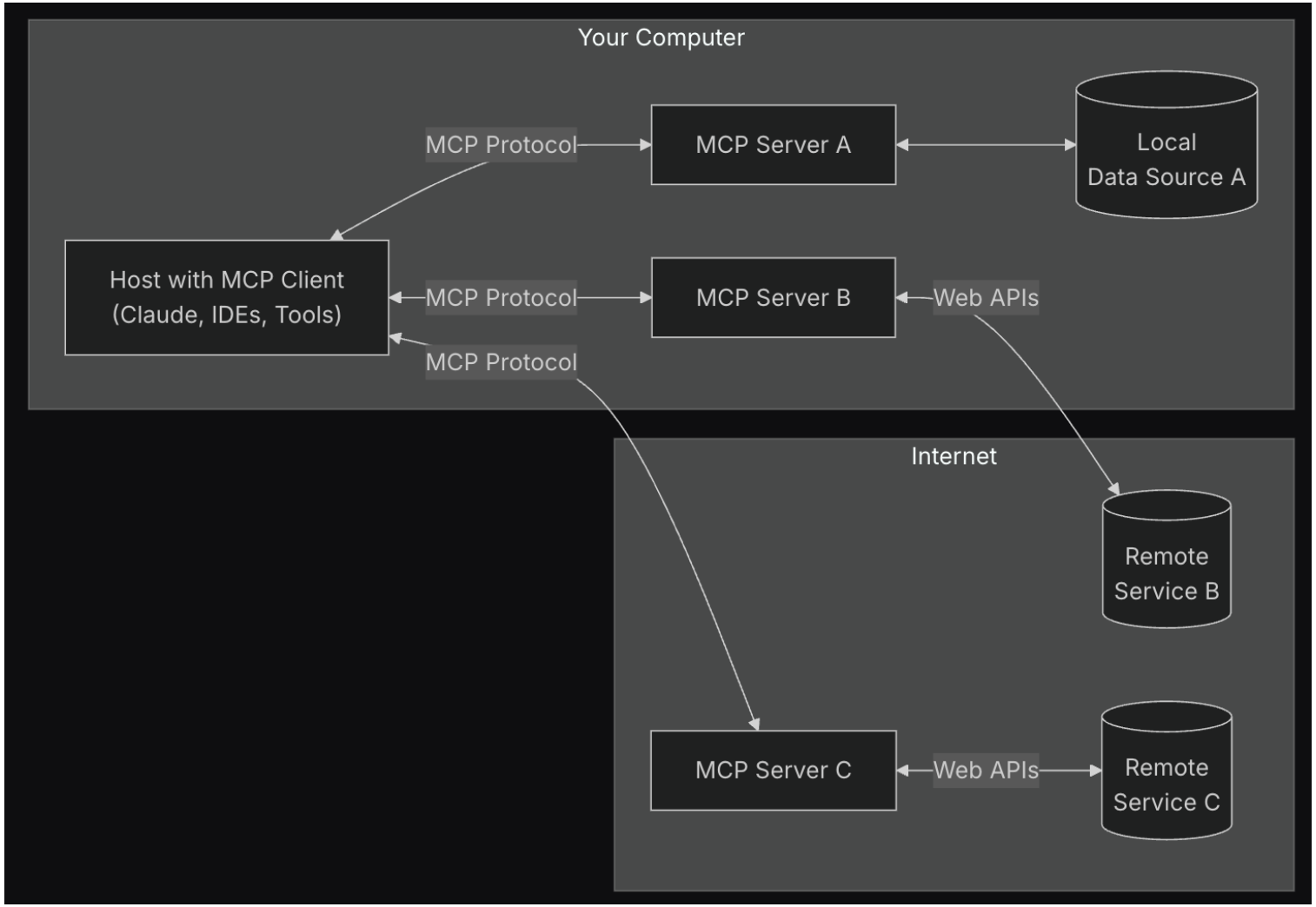
API vs MCP

- Very similar in concept
- APIs have “endpoints”, MCP servers have “tools”
- Different use cases but similar implementations
- Different architectural decisions
- Different documentation
- Can convert from one to the other
- Should like re-use a significant amount of code and infrastructure between an MCP server and a well written API

MCP Concepts

- MCP is a protocol that exposes custom code to LLMs
- Think of MCP as providing “context” or “tools” to your LLMs or Agents (or other AI enabled applications)
  - AI calls your code when needed
  - AI determines inputs and evaluates output
- A single AI prompt might be evaluated by the AI LLM and the decision made to call many MCP tools (from the same or different endpoints). This chaining is very powerful and affects your design.
- In a local C#/dotnetcore application you can register tools in memory via SemanticKernel SDK…remote tools are registered and called via MCP.

MCP Server Architecture



Describing MCP Tools (in C#)

- Static Classes (per the current SDK)
- C# Attributes
  - Class Level
  - Method Level
  - Parameter Level
  - Model Level (inputs/outputs)
- Easy to scan and expose MCP with very little up-front configuration

- Tooling:
  - Postman

Building an MCP Server

- SDKs in multiple languages
- Tools are descriptive endpoints made available to LLMs
- Very easy to write and describe in most languages
- Deployment can be:
  - Local (executable or command line)
  - Remote (streaming HTTP)

Building an MCP server in C#

```
builder.Services.AddMcpServer()
    .WithHttpTransport()
    .WithToolsFromAssembly();

…

app.UseEndpoints(endpoints =>
{
    endpoints.MapControllers();
    endpoints.MapMcp("/mcp");
    endpoints.MapRazorComponents<App>()
        .AddInteractiveServerRenderMode();
});
```

Modern Applications

- User Interface
  - Angular, React, Blazor, Mobile, Other?
- API
  - Json/Rest
- MCP
  - Tools

Lots of options but will commonly be a single versionable deployment. All hosted in the Cloud.

UI will become less complex in certain areas due to MCP

Demos

Use Cases

- Integrate Existing or new Apps with AI Agents
- Wrap API's in MCP endpoints

- Examples of apps now exposing MCP Server capabilities:

  - GitHub
  - Azure
  - Azure DevOps
  - Figma
  - Windows (or other operating systems)
  - 100s of others (soon to be 1000's)

[VS Code MCP Servers]

Final Thoughts

How do we search or use the web today?

How do we utilize custom applications?

How will we do that tomorrow?

Most applications will support registering MCP servers to expand capabilities

Interaction between websites, local applications, etc will continue to be driven by LLMs and MCP

Resources

- MCP Servers in VS Code
- Building MCP Servers in C#
- GitHub MCP Server
- C# MCP SDK
- Official MCP Documentation
- Oath in MCP (Blog Post)
- Figma MCP Server

THANK YOU

Kevin Grossnicklaus
kvgros@architectnow.net
www.architectnow.net
GitHub: kvgros2