Lesson: Introduction to the Model Context Protocol (MCP)

What is MCP?

Welcome! In this lesson, we're diving into **MCP**: the **Model Context Protocol**. MCP is an **open protocol** designed to *standardize connections* between large language models (LLMs) and the context they use.

But what does this actually mean?
Let's explore how MCP changes the way we build AI agents, especially looking ahead to 2025.

The Modern Agent Loop

Imagine creating an agent application:

1. **User sends a prompt** to the LLM.
2. The LLM can:
   - Use tools (like web search, calendar, email).
   - Send results back to the user.
3. The LLM may loop through these steps several times until the task is done.

This back-and-forth is called the **agent loop**. It's a simple, but powerful, structure behind most agent applications today.

What's the Challenge?

To build an agent like this, you need:

- **LLM API access** (to interact with the model itself),
- **Connections to tools/resources** (e.g., search, calendar, etc.),
- **Agent logic** (to coordinate actions).

Other production needs (like orchestration, monitoring, observability, evaluation) also exist, but let's focus on these core components for now.

The M $\times$ N Integration Problem

Here's where it gets difficult:

Suppose you want to support **multiple LLMs** (maybe from different providers). Each LLM might need to connect to **all the same tools and resources**, but each does so differently.
This means you have to **rebuild integrations for each LLM**, for each tool.

- If you have **M** models and **N** integrations, you end up with **M** $\times$ **N** separate connection setups!

This leads to: - Redundant work. - Complex, hard-to-maintain codebases.

Enter MCP: Solving the Integration Problem

MCP acts as a standard "middle layer" between LLMs and all their tools, resources, and prompts (collectively called "context").
With MCP:

- Each LLM connects **once to the MCP**.
- Each tool/resource also connects **once to the MCP**.

Now, instead of **M** $\times$ **N** integrations, you just need **M** + **N** integrations.

MCP Infrastructure — The Core Components

Let's break down what MCP is composed of:

1. **Host**
   - The user-facing application (e.g. a cloud app, desktop app, or IDE).
2. **Client**
   - A sub-component within the Host.
   - Handles one-to-one connections to MCP Servers.
3. **MCP Server**
   - Exposes tools, resources, and prompts to the Clients.
   - Allows Clients to:
     - Discover available capabilities.
     - Relay these to the embedded LLM in the Host.

With this, the **LLM can choose** which tools/resources to use for user tasks—standardized, dynamic, and extendable.

Summary

- MCP **standardizes** integrations between LLMs and their context.
- It **drastically reduces** the engineering overhead for multi-model, multi-resource agents.

- Its architecture (Hosts, Clients, MCP Servers) makes tool/resource integration easy and scalable.

Next Steps

In the **next lesson**, we'll take a closer look at the core components of MCP and see how they interact in real applications.

See you there!