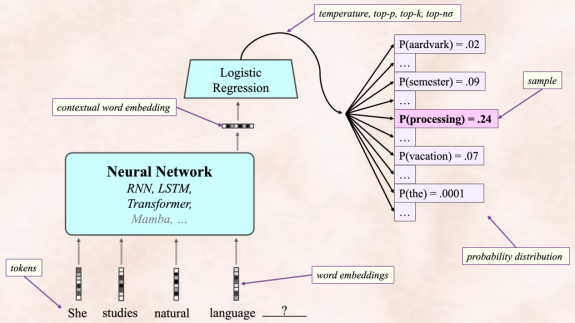
ITCS 4101: Introduction to NLP

Model Context Protocol (MCP)

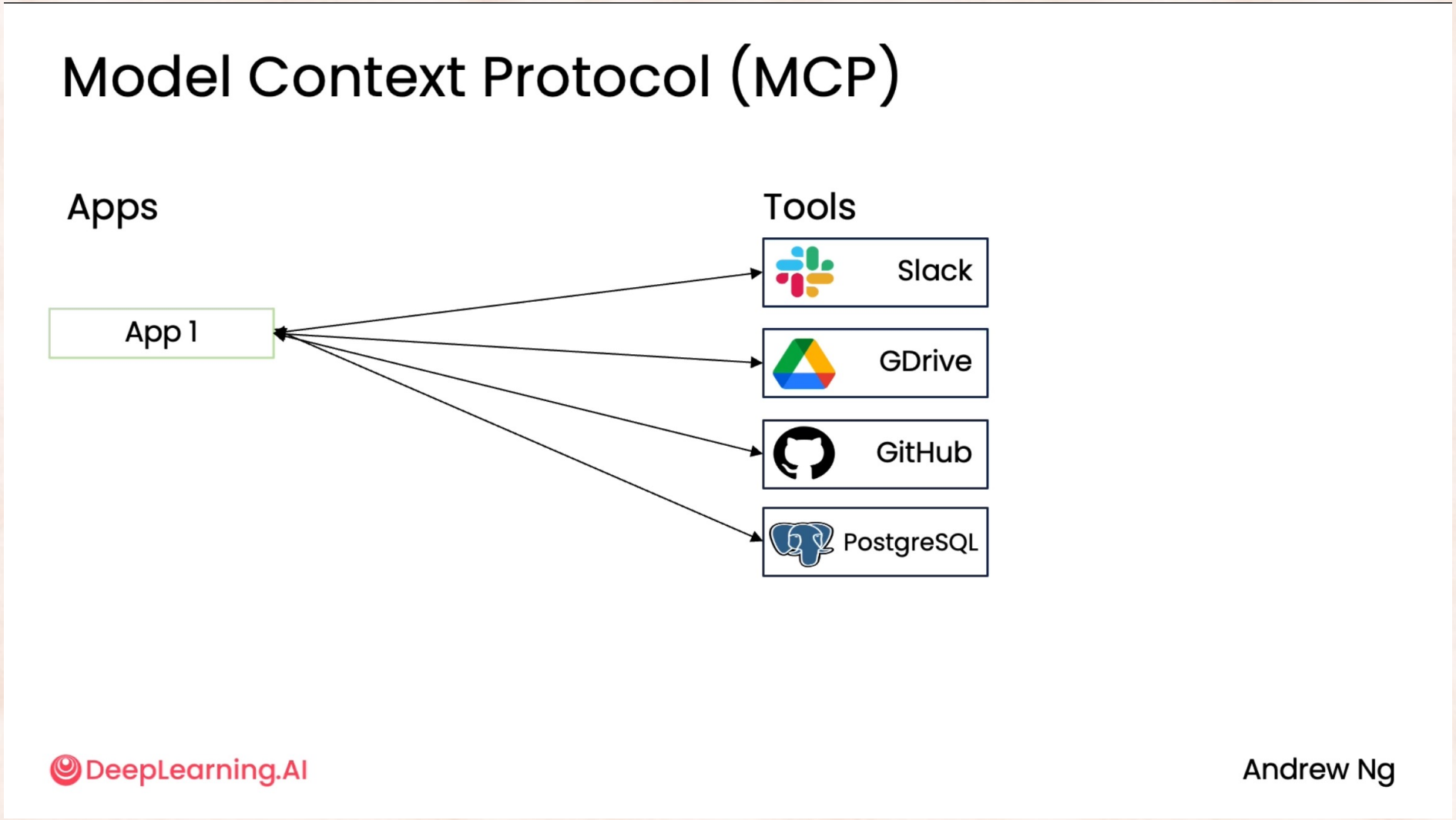


Razvan C. Bunescu

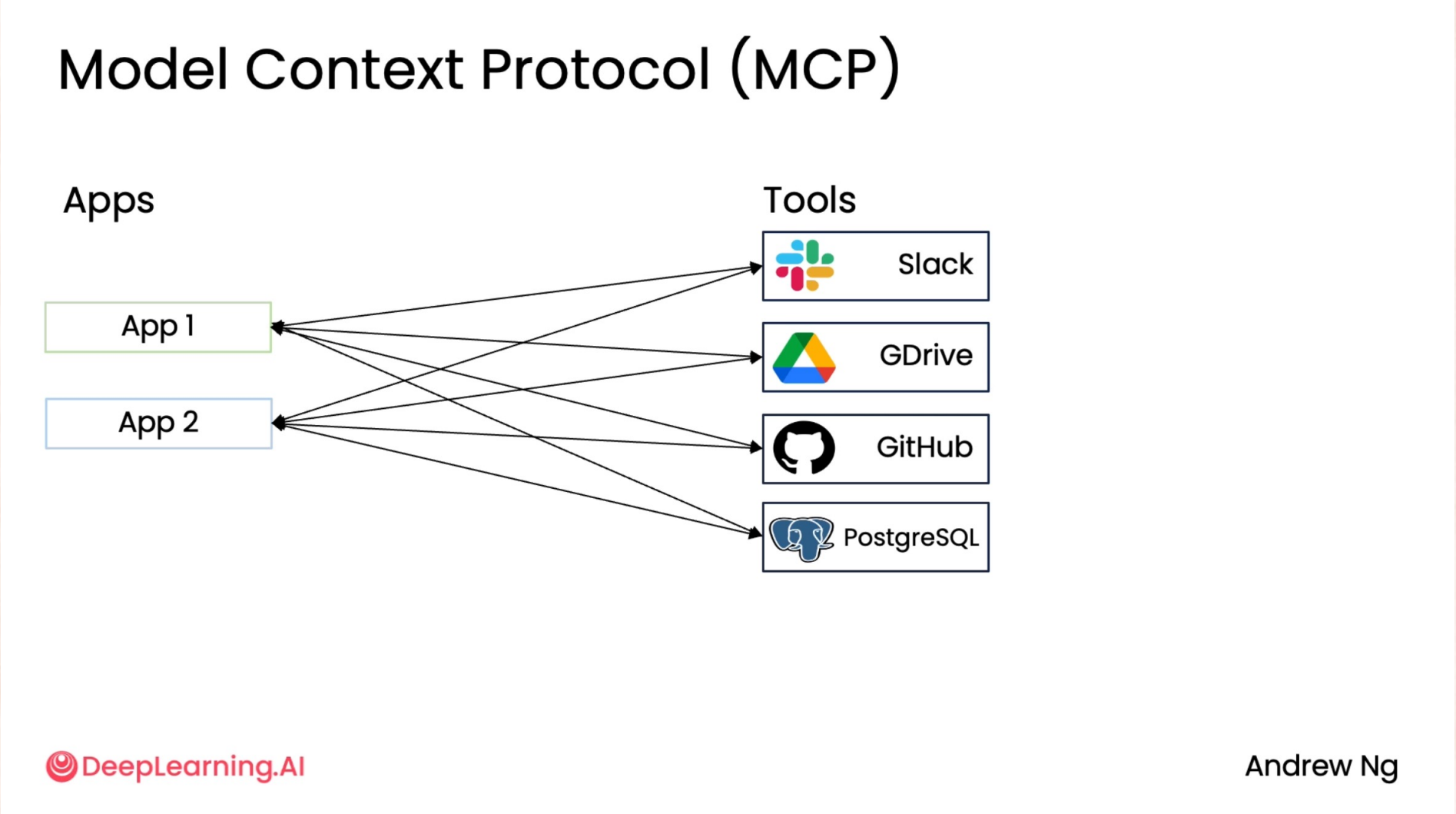
Department of Computer Science @ CCI

rbunescu@charlotte.edu

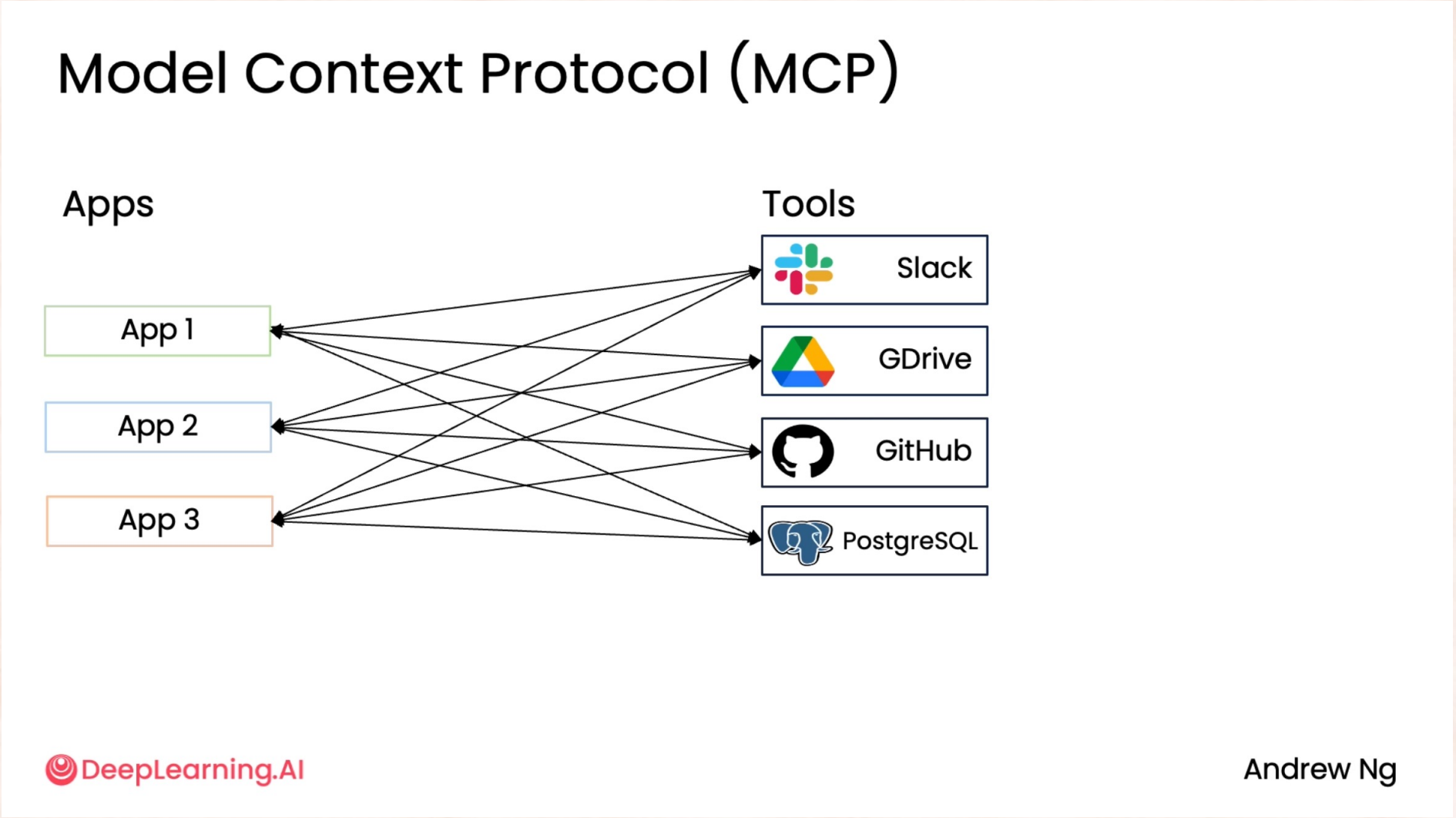
What does MCP solve?



What does MCP solve?

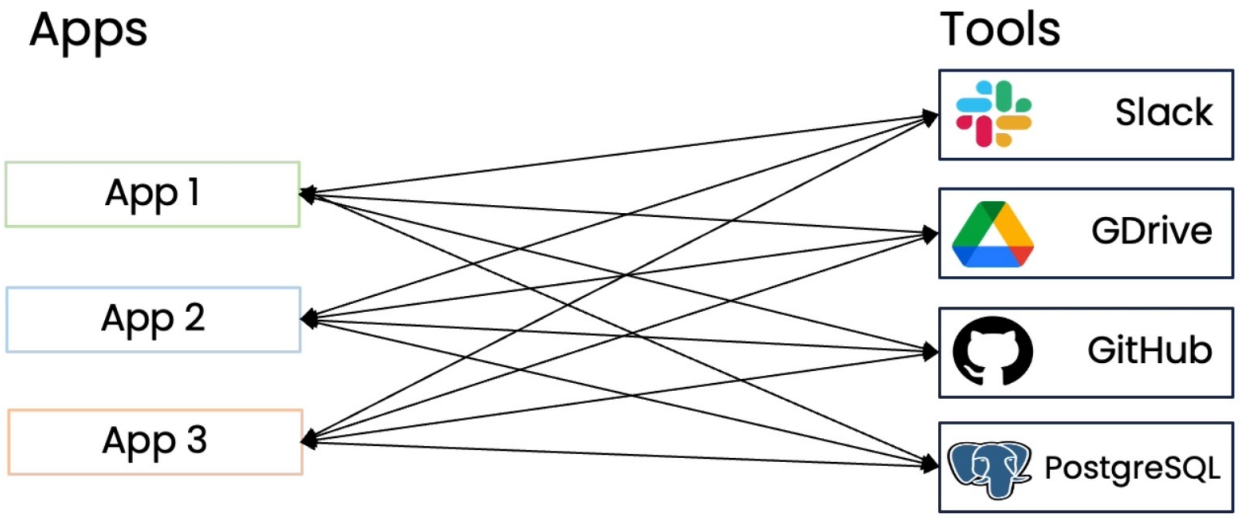


What does MCP solve?



What does MCP solve?

Model Context Protocol (MCP)



w/o MCP

Each app creates their own tools

$m \times n$

w/ MCP

Each app uses shared MCP server

$m + n$

DeepLearning.AI

Andrew Ng

What does MCP solve?

Using pre-built clients and servers

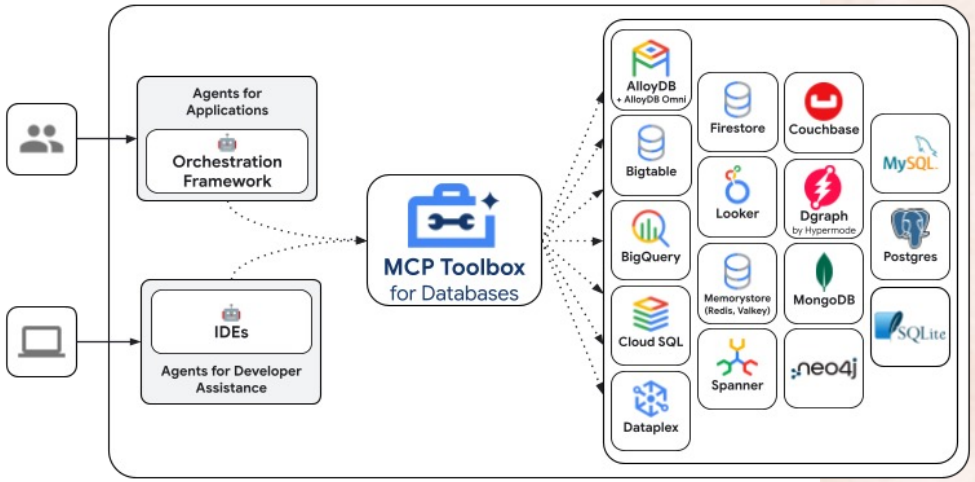
Clients

- Cursor
- Claude Desktop
- Windsurf
- Your App

Servers

- Slack
- Google Drive
- GitHub
- PostgreSQL

Many servers available, some developed by the service providers.



https://google.github.io/adk-docs/mcp

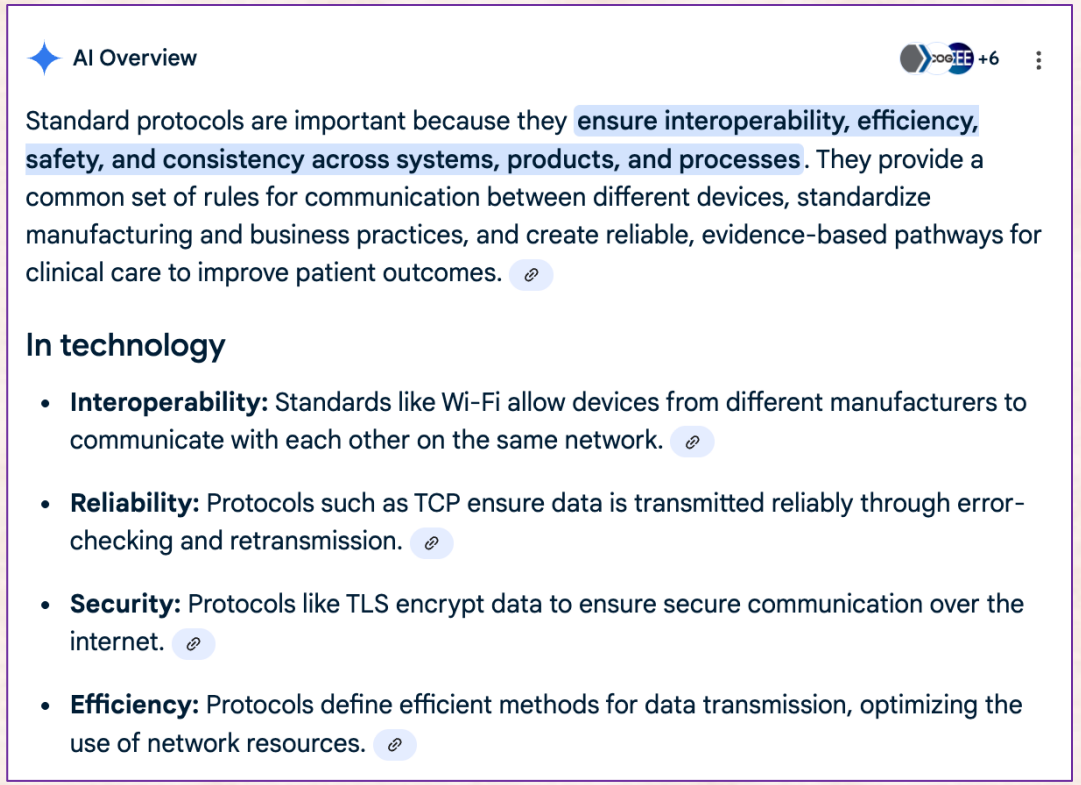
DeepLearning.AI

What is MCP?

https://modelcontextprotocol.io

Model Context Protocol (MCP) is an **open-source standard** for **connecting** **AI applications** to:

- **Tools**: Executable functions that AI applications can invoke to perform actions:
  - File operations, API calls, database queries, …
- **Resources**: Data sources that provide contextual information to AI applications:
  - File contents, database records, …
- **Prompts**: Reusable templates that help structure interactions with language models:
  - System prompts, few-shot examples, …



The MCP creates a standardized, two-way connection for AI applications, allowing LLMs to easily connect with various data sources and tools. MCP builds on existing concepts like tool use and function calling but standardizes them. This reduces the need for custom connections for each new AI model and external system. It enables LLMs to use current, real-world data, perform actions, and access specialized features not included in their original training.
https://cloud.google.com/discover/what-is-model-context-protocol

Some Standard Protocols

- **Model Context Protocol (MCP)**:
  - An open, JSON-RPC based protocol that allows large language models (LLMs) to connect to external tools and data sources, such as APIs and databases.
- **Language Server Protocol (LSP):**
  - An open, JSON-RPC based protocol for use between source-code editors or integrated development environments (IDEs) and servers that provide "language intelligence tools": programming language-specific features like code completion, syntax highlighting and marking of warnings and errors, as well as refactoring routines.
- **Hypertext Transfer Protocol (HTTP):**
  - An application-layer protocol that operates on a client-server model, allowing web browsers and other clients to request and receive resources from web servers.
- **Open Authorization (OAuth):**
  - Open standard protocol for authorization. It allows a third-party application to access a user's data on another service without sharing the user's password.

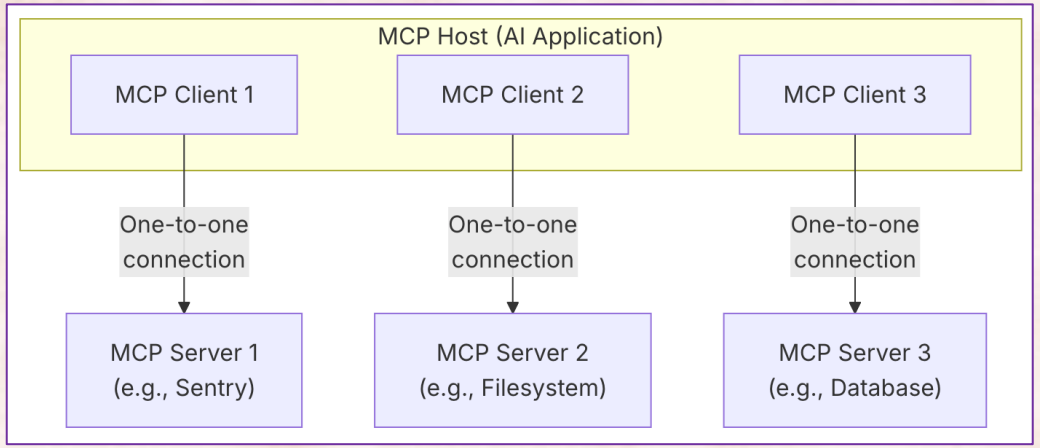
MCP Data and Transport Layers

- Model Context Protocol layers:
  - **Data layer** implements a JSON-RPC 2.0 based exchange protocol that defines the message structure and semantics.
    - **Lifecycle management**: connection initialization / termination, capability negotiation.
    - **Server features**: Enables providing core functionality: **tools**, **resources**, and **prompts**.
    - **Client features**: Enables servers to ask the client to sample from the host LLM, elicit input from the user, and log messages to the client.
    - **Utility features**: Notifications for real-time updates and progress tracking for long-running operations
  - **Transport layer** manages communication channels and authentication between clients and servers:
    - **Stdio**: Uses standard input/output streams for direct process communication between local processes on the same machine, providing optimal performance with no network overhead.
    - **Streamable HTTP**: Uses HTTP POST for client-to-server messages with optional Server-Sent Events for streaming capabilities. This transport enables remote server communication and supports standard HTTP authentication methods including bearer tokens, API keys, and custom headers.
      - MCP recommends using **OAuth** to obtain authentication tokens.

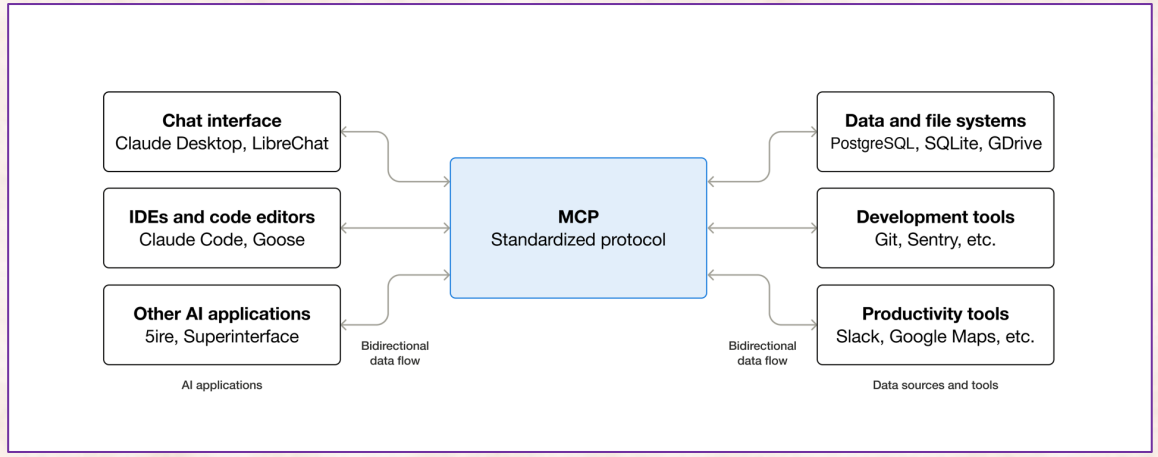
https://modelcontextprotocol.io/docs/learn/architecture

MCP Architecture and Key Components

MCP Architecture: **hosts**, **clients**, **servers**



Example MCP compliant client-server communication



- **MCP Host**: The LLM is contained within the MCP host, such as an AI-powered IDE or conversational AI:
  - This is typically the user's interaction point, where the MCP host uses the LLM to process requests that may require external data or tools.
- **MCP Client**: Located within the MCP host, it helps the LLM and MCP server communicate with each other.
  - It translates the LLM's requests for the MCP and converts the MCP's replies for the LLM; also finds and uses MCP servers.
- **MCP Server**: The external service that provides context, data, or capabilities to the LLM:
  - It helps LLMs by connecting to external systems like databases and web services, translating their responses into a format the LLM can understand which helps developers provide diverse functionalities.

Python MCP Development with FastMCP

- FastMCP is a Python framework for building MCP client-server applications.
  1. Example from Anthropic:
     - Build an **MCP weather server** that uses data from the National Weather Service API and exposes two tools: get_alerts() and get_forecast().
       - Then connect the server to an **MCP host**, e.g., Claude for Desktop.
       - https://modelcontextprotocol.io/docs/develop/build-server
     - Build an **MCP client** that connect to the MCP weather server and lists available tools, starts an interactive chat session: Enter queries, See tool executions, Get responses from Claude LLM.
       - https://modelcontextprotocol.io/docs/develop/build-client
  2. Example from FastMCP:
     - Build a simple MCP server that exposes toy tools, such as multiply() and greet().
       - Running the server, specifying the protocol, authentication with OAuth.
     - Build simple MCP clients that connect to one or multiple servers.
       - Configuration, list tools, authentication with OAuth.

My MCP Client Server Examples

1. Python MCP server uses the Yelp *business.json* database:
   - It will expose two tools:

```
from fastmcp import FastMCP

mcp = FastMCP("My MCP Restaurant Server")
```

```
@mcp.tool
def get_restaurants_for_location_cuisine(city: str, state: str, cuisine: str) -> list:
    """Gets a list of restaurants that are located at a particular location (city and state) and serve a particular cuisine.
    """
```

```
@mcp.tool
def is_restaurant_kid_friendly(name, full_address):
    """Returns True if a restaurant is kid friendly, False otherwise.
    """
```

2. Python MCP host defines an MCP client that tests the tools:
   - A second client connect to a remote server.

```
import asyncio
from fastmcp import Client
import json

client = Client("http://localhost:8000/mcp")

async def call_tools(city, state, cuisine):
    async with client:
```

3. Notebook MCP host create a Gemini client and defines an MCP client that acts as a bridge between Gemini and the MCP server.

My MCP Client Server Examples

- The `llm-mcp/fast-mcp/` folder contains:
  - My MCP server implementation in `yelp_server.py`
  - The MCP host with two clients in `yelp_host.py`
  - A weather server based on the NWS API in `weather_server.py`
  - A notebook that uses an LLM + MCP client for the local server, in `MCPClientLLM.ipyng`

1. Install the fastmcp module:
   ```
   pip install fastmcp
   ```
2. Start the server in a terminal window:
   ```
   fastmcp run yelp_server.py:mcp --transport http --port 8000
   ```
3. Run the MCP host in another terminal window:
   ```
   python yelp_client.py
   ```

MCP Examples

- More FastMCP examples.

- A collection of reference and third party MCP servers.
  - Everything - Reference / test server with prompts, resources, and tools.
  - Fetch - Web content fetching and conversion for efficient LLM usage.
  - Filesystem - Secure file operations with configurable access controls.
  - Git - Tools to read, search, and manipulate Git repositories.
  - Memory - Knowledge graph-based persistent memory system.
  - Sequential Thinking - Dynamic and reflective problem-solving through thought sequences.
  - Time - Time and timezone conversion capabilities.
  - and many more …

Supplemental Materials

- MCP: Build Rich-Context AI Apps with Anthropic:
  - Short course taught by Elie Schoppik from Anthropic, distributed by deeplearning.ai.
- Google's MCP Toolbox For Databases.
- **Asynchronous programming** in Python with asyncio:
  - asyncio tutorial from the BBC R&D Cloudfit team:
    - part 1, part 2, part 3, part 4, part 5, …
  - A Conceptual Overview of asyncio.

This one!