MCP: Every Developer Needs to Know About

2 min read · Jun 2, 2025

Ad Tech Pulse Follow

What is MCP?

Model Context Protocol (MCP) is changing how AI models connect with external data sources. Think of it as the USB-C of artificial intelligence. One standard protocol for all connections.

Before MCP, connecting AI to databases was messy. Each integration needed custom code. Different APIs meant different authentication methods. Developers spent countless hours building fragile connections.

MCP solves this chaos with a unified approach.

Background / Problem Statement

Early AI models like ChatGPT were impressive but limited. They could only work with their training data. No real-time information. No access to your personal files.

Later it started utilizing your personal data in the same context and then hitting LLM requests. Perplexity combined LLMs with web search. Cursor integrated AI with your codebase. Google Gemini connected with your workspace apps. So, these weren't just chatbots anymore.

Success brought new challenges. Every AI app needed custom connections to different tools. PostgreSQL required specific code. Slack needed different authentication. File systems demanded another approach entirely. This created an **N-to-M problem.** Multiple AI apps trying to connect with multiple data sources. Each connection was unique and fragile.

How MCP Works

MCP introduces three key components

- **MCP Client:** Your AI application that needs data. This runs inside tools like Cursor or custom AI apps.
- **MCP Server:** A lightweight program that speaks to specific services using MCP standards. Need database access? Run an MCP server for that database.
- **Data Sources:** The actual systems you want to connect. Local files, remote APIs, cloud databases, or web services.

Simple Python Example

Before MCP

```python
import sqlite3
import openai

def get_user_data(user_id):
    conn = sqlite3.connect('users.db')
    cursor = conn.cursor()
    cursor.execute('SELECT * FROM users WHERE id = ?', (user_id,))
    result = cursor.fetchone()
    conn.close()
    return result

def ask_ai_about_user(user_id):
    user_data = get_user_data(user_id)
    prompt = f"Analyze this user data: {user_data}"
    response = openai.ChatCompletion.create(
        model="gpt-4",
        messages=[{"role": "user", "content": prompt}]
    )
    return response.choices[0].message.content
```

After MCP

```python
from mcp import Client

client = Client("http://127.0.0.1:8000/mcp")
async with client:
    result = await client.call_tool("analyze_user", {
        "query": "Can you analyze user_id = 123?"
    })
    print(result)
```

The Future Vision

Imagine telling your AI assistant: “Check my database for user engagement metrics, create a summary report, and post it to our Slack channel.”

With MCP, this becomes possible without complex integration code. The AI orchestrates across multiple systems using standardized connections.

Getting Started Today

Start small with one MCP server. Maybe connect your AI tool with local files or a simple database. Experience how much cleaner the integration feels.

Popular MCP servers already exist for:

- GitHub repositories
- PostgreSQL databases
- Slack workspaces
- Local file systems
- Stripe payments
- Notion databases

The Bottom Line

MCP represents a fundamental shift in AI integration. Instead of building custom bridges for every connection, we get a universal standard. This isn’t just another developer tool. It’s infrastructure for the AI-powered applications we’re building today and tomorrow.

The question isn’t whether MCP will become important. It’s whether you’ll adopt it early enough to benefit from the simplified development experience.

Llm Model Context Protocol Mcp Server Mcp Client

Follow

Written by Ad Tech Pulse

1 follower · 4 following

Passionate

No responses yet

Bgerby

What are your thoughts?

More from Ad Tech Pulse

Ad Tech Pulse

Implementing Python Decorators as Method Annotations

Introduction

Nov 22, 2023

Ad Tech Pulse

Building an Automated Data Entry Application with Google Sheets Add-Ons

Introduction

Nov 22, 2023

Ad Tech Pulse

Unit Tests and Code coverage with Golang

This is a simple example to illustrate how unit tests can be written in Golang. This article also explains how a code coverage percentage...

May 8, 2020 2

Ad Tech Pulse

An Introduction to Google Cloud Functions: Building Scalable and Secure Serverless Applications

Introduction

Dec 4, 2023

See all from Ad Tech Pulse

Recommended from Medium

Shivee Gupta

Building a Production-Grade Observability Platform with SigNoz, ClickHouse, and OpenTelemetry —...

Lessons from our in-house observability setup and what we learned beyond the documentation and long nights of cluster tuning.

Oct 27 46 1

Heeki Park

Building an MCP server as an API developer

Anthropic released MCP at the end of November 2024. It took a few months for it to catch on, but my, oh my, it feels like the community is…

May 14 577 5

In AI Software Engineer by Joe Njenga

Anthropic Just Solved AI Agent Bloat — 150K Tokens Down to 2K (Code Execution With MCP)

Anthropic just released smartest way to build scalable AI agents, cutting token use by 98%, shift from tool calling to MCP code execution

4d ago 417 33

Sofiane Ghadab

Bridging MCP Transports: Turning an STDIO Server into SSE (and the Other Way Around)

Model-Context Protocol (MCP) servers come in two transport flavors:

Jul 11 15

CodeOrbit

I Built a RAG System for 100,000 Documents—Here's the Architecture

My production system crashed at 2 AM because I underestimated vector databases.

Nov 1 583 15

In Medialesson by Marius Schröder

JSON vs TOON—A new era of structured input?

Why structure matters more than ever

Nov 3 163 9

See more recommendations