Before Starting

The Why

The What

The How

The Problem Statement

Learning in the age of AI

Our own newsletter

Time Constraints

Step 1 - The Structure

Introduction

Big Story of the Week

3–5 Quick Updates

Top Research Papers

Top Github Repos

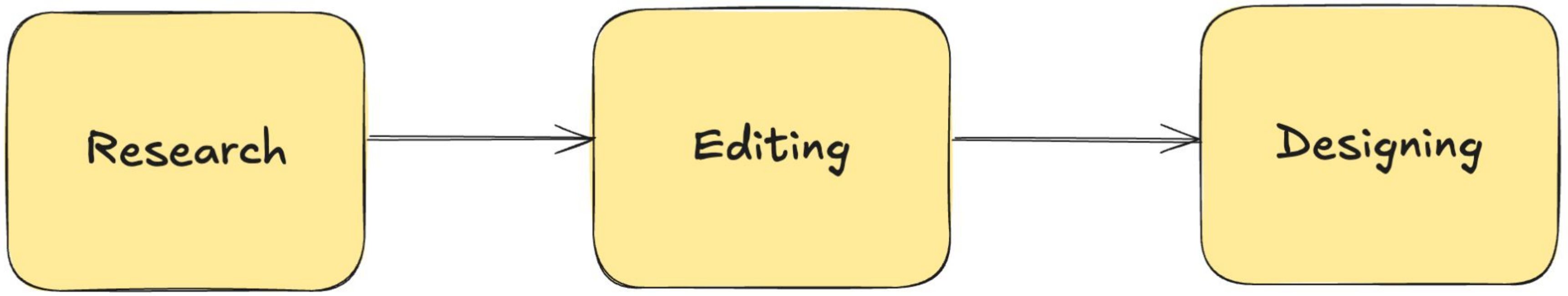
1 Quick Tutorial

Top AI products

Top X posts

Closing notes

Step 2 - Defining the Process



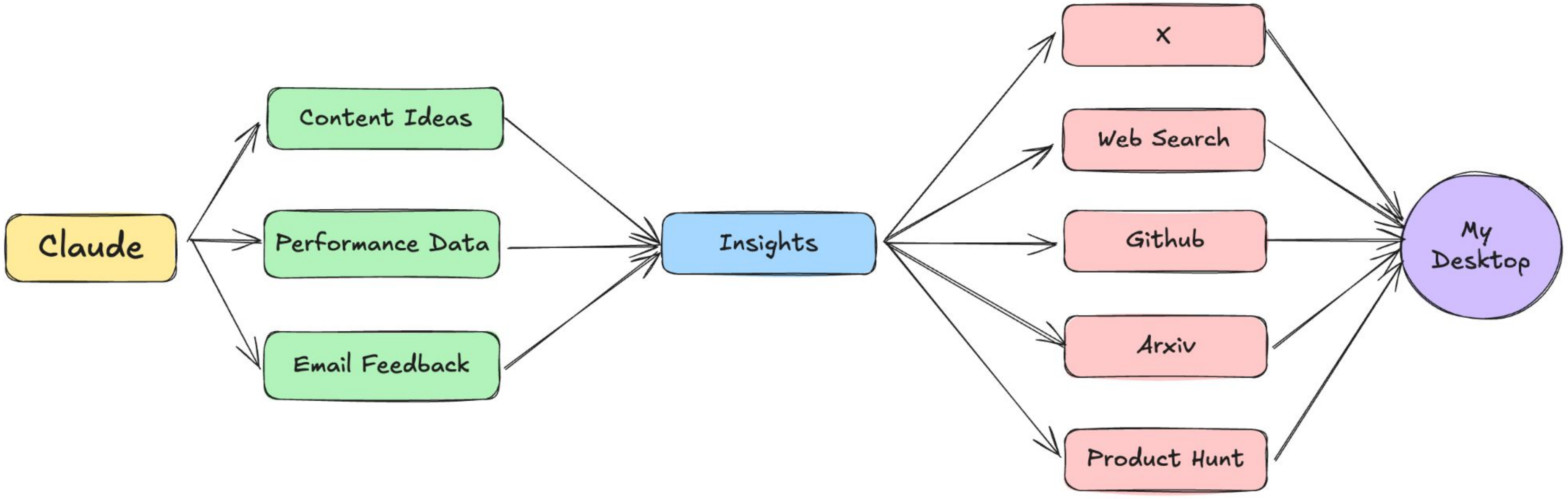
Step 3 - Deciding the Tools

The Main AI

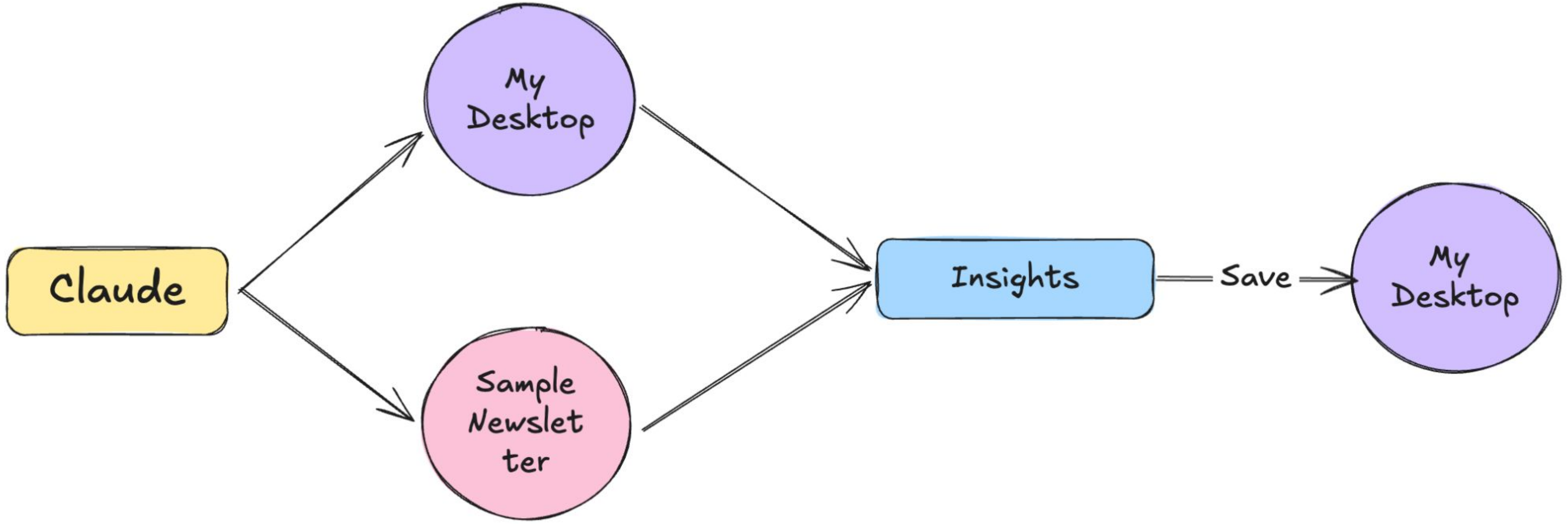
The Tools

Github	Web Search	Google Drive
Arxiv	Gmail	Product Hunt

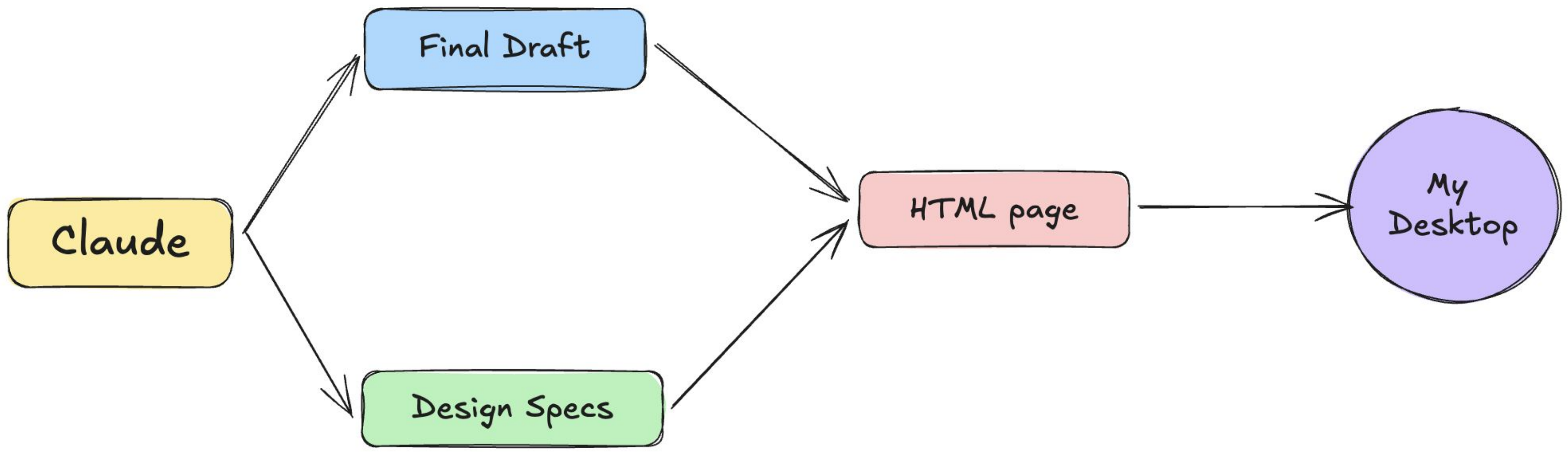
Step 4 - The Research Phase



Step 5 - Editing Phase



Step 6 - Designing Phase



PART 1

THE WHY

The Arrival of LLMs

ChatGPT was launched on Nov 30 2022

It crossed 1 Million users in 5 days

Then it crossed 100 Million users in 2 months

Not another software launch.

Waves of Adoption

Wave 1 - Pure Wonder

Explain Quantum Physics from a cat’s perspective

What would happen if gravity worked backwards

Write a song about pizza in the style of Shakespeare

Impact - Social Media Exploded

Waves of Adoption

Wave 2 - Professional Adoption

Lawyers: "Summarize this 50-page contract."

Developers: "Debug this Python function."

Teachers: "Create a lesson plan about photosynthesis."

Impact - Individual productivity boom

Waves of Adoption

Wave 3 – The API Revolution

Copilot across Word, Excel, PowerPoint, and Outlook

AI in Gmail, Docs, Sheets, and Drive

New AI-first tools like Cursor, Perplexity emerged

Impact - AI became more accessible

The Problem of Fragmentation

AI in notion couldn't talk to AI in Slack

VS Code coding assistant knew nothing about discussions in MS teams

People found themselves living in multiple AI worlds

Users were juggling between multiple AI assistants

The Vision vs The Reality

They wanted one unified AI partner that can understand their work

A unified tool that can solve any problem related to their work

Users never wanted 5 different AI tools

But there was a big problem in building a unified AI

The problem of Context

What is Context

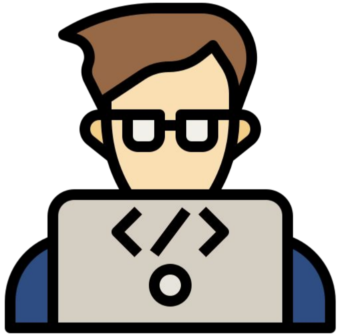
Context is everything an AI can "see" when it generates a response.

More formally, Context refers to the information(conversation history, external docs etc) that the LLM uses to generate a response.

For e.g. while chatting with ChatGPT, the past messages forms the context.

Example - Software Engg.

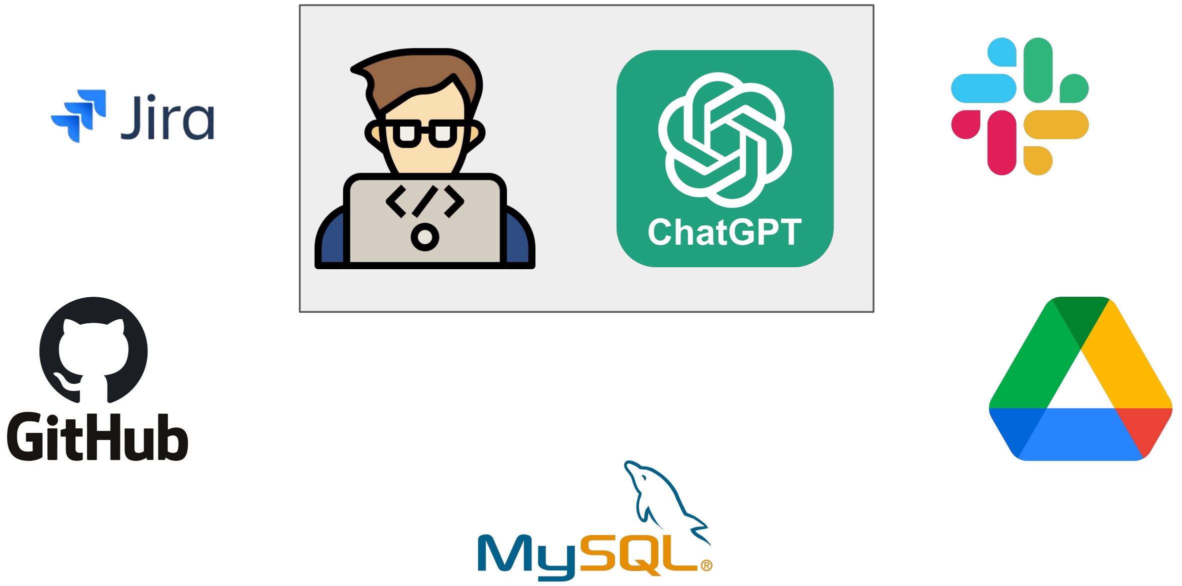
Jira



GitHub

MySQL®

Software Engg. workflow with AI



Context is scattered

The Copy-Paste Hell

Need to paste thousands of line to ask one simple question

Developers have become Human APIs

Context Assembly Time > Development Time

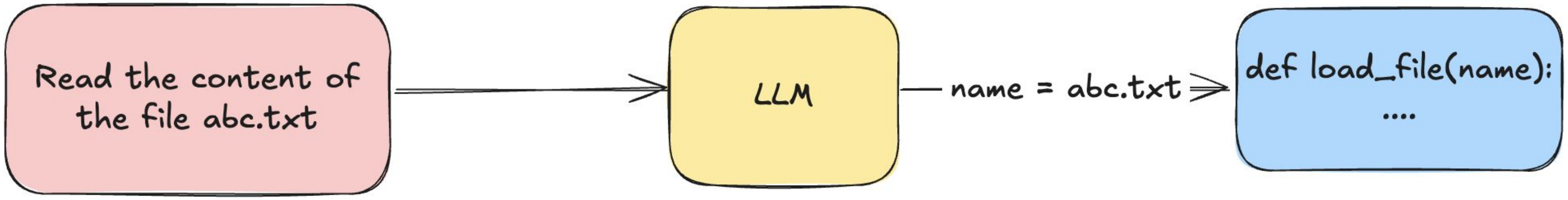
Managing what the AI remembers

Scaling problems

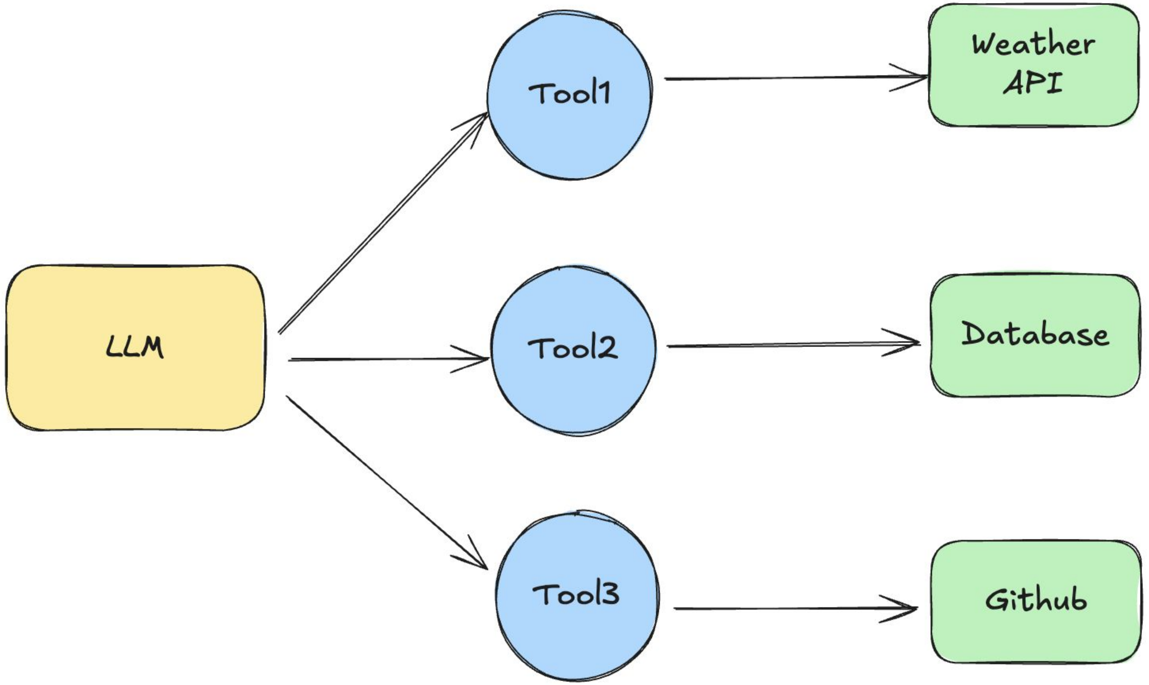
The Solution - Function Calling

OpenAI introduced function calling in mid 2023

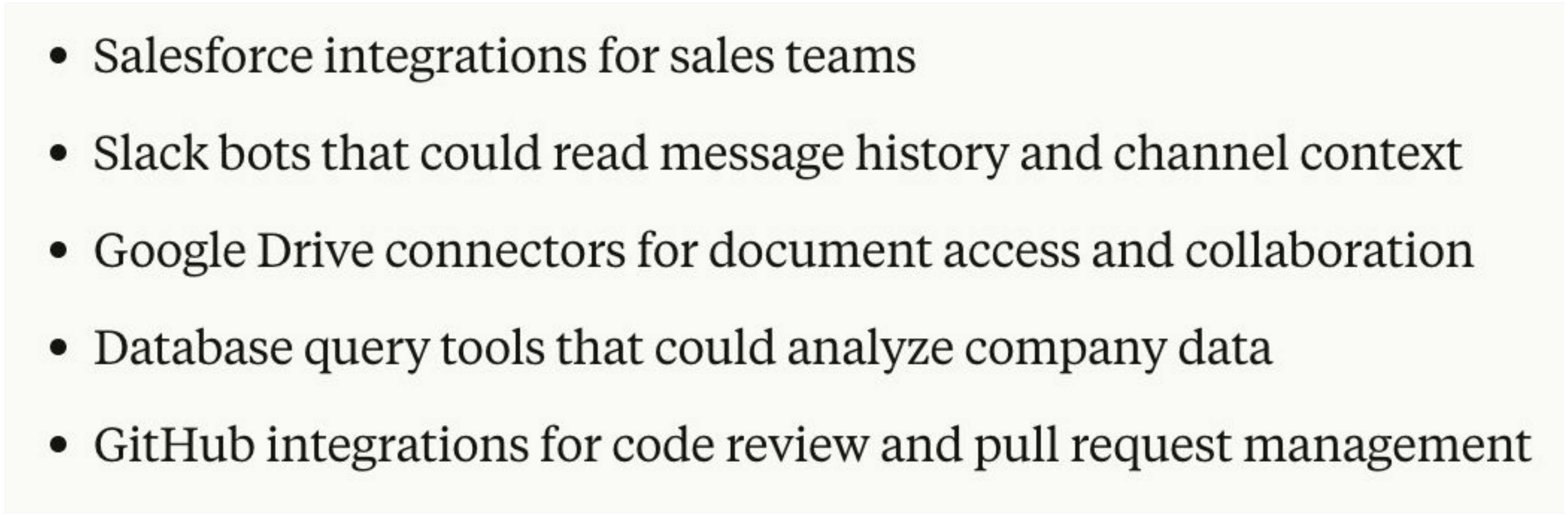
Function Calling is a way using which LLMs can call ext functions



The Rise of Tools

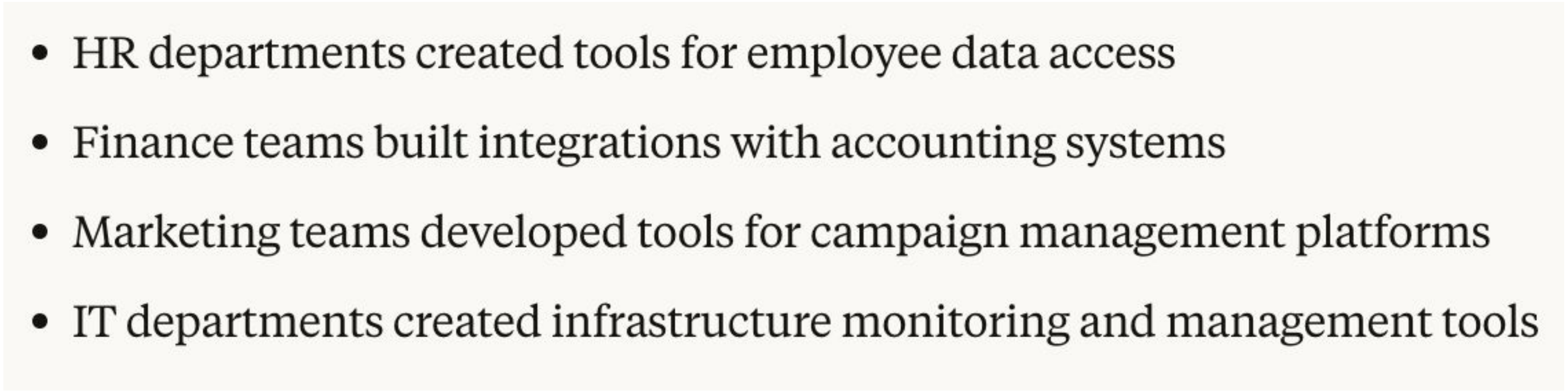


Implication of Tools

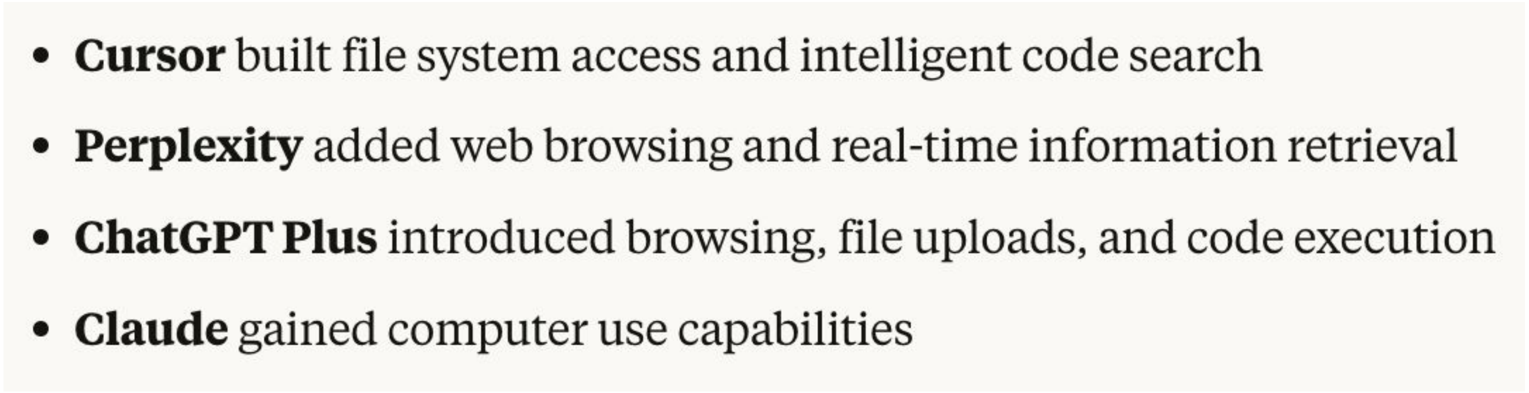


- Salesforce integrations for sales teams
- Slack bots that could read message history and channel context
- Google Drive connectors for document access and collaboration
- Database query tools that could analyze company data
- GitHub integrations for code review and pull request management

Implication of Tools

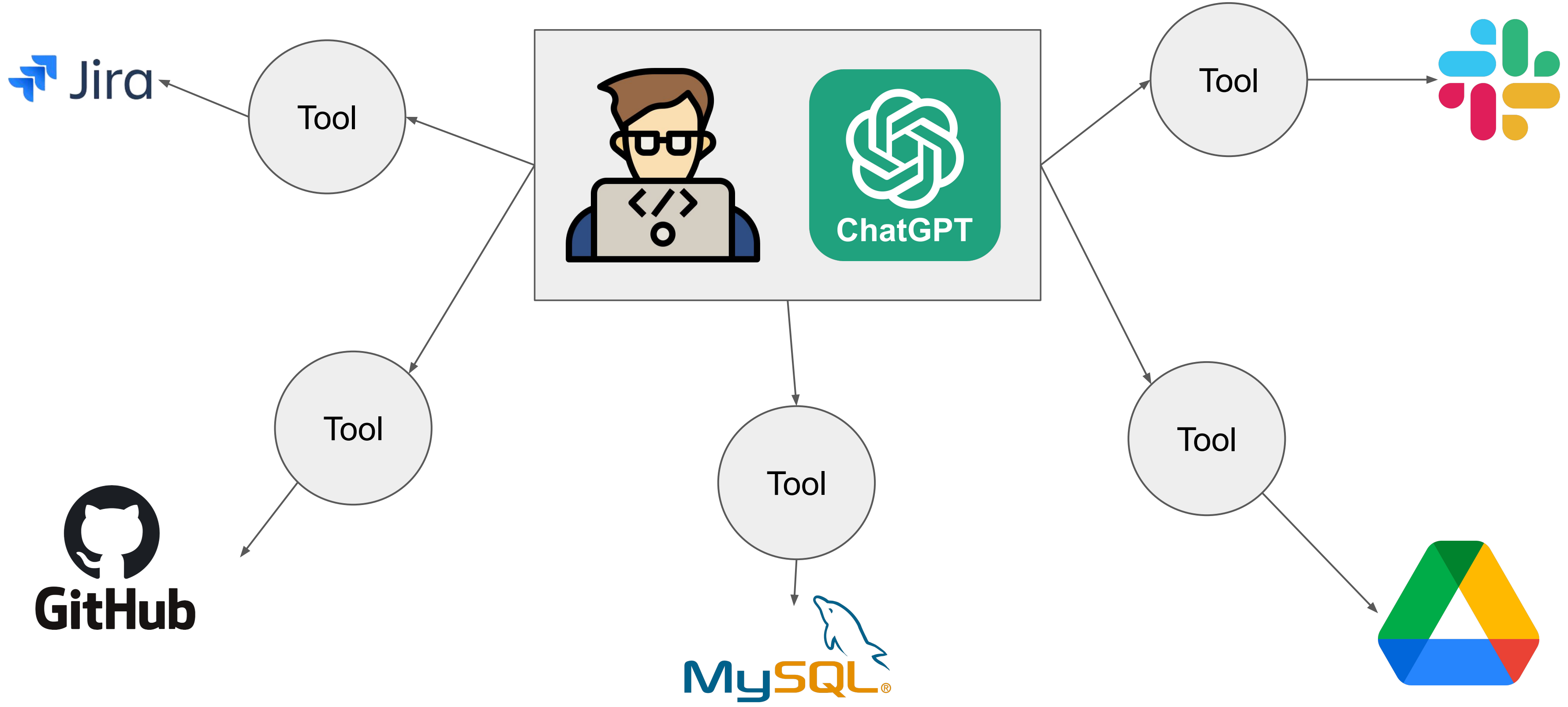


- HR departments created tools for employee data access
- Finance teams built integrations with accounting systems
- Marketing teams developed tools for campaign management platforms
- IT departments created infrastructure monitoring and management tools



- **Cursor** built file system access and intelligent code search
- **Perplexity** added web browsing and real-time information retrieval
- **ChatGPT Plus** introduced browsing, file uploads, and code execution
- **Claude** gained computer use capabilities

The New Scenario



Problem with Tools

Integration Problem

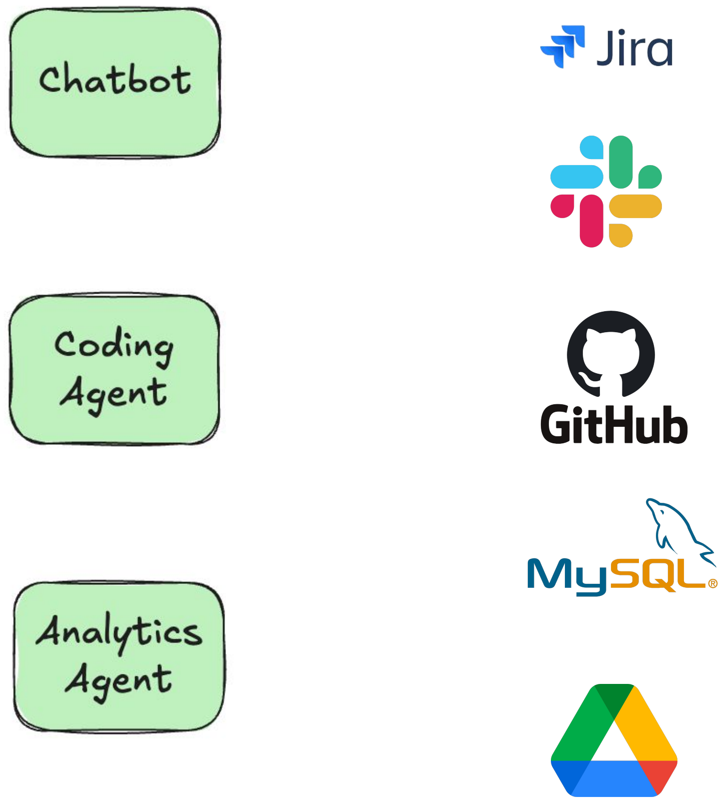
N * M

Development Nightmare

Diff authentication methods

Diff data formats and API patterns

Diff error handling



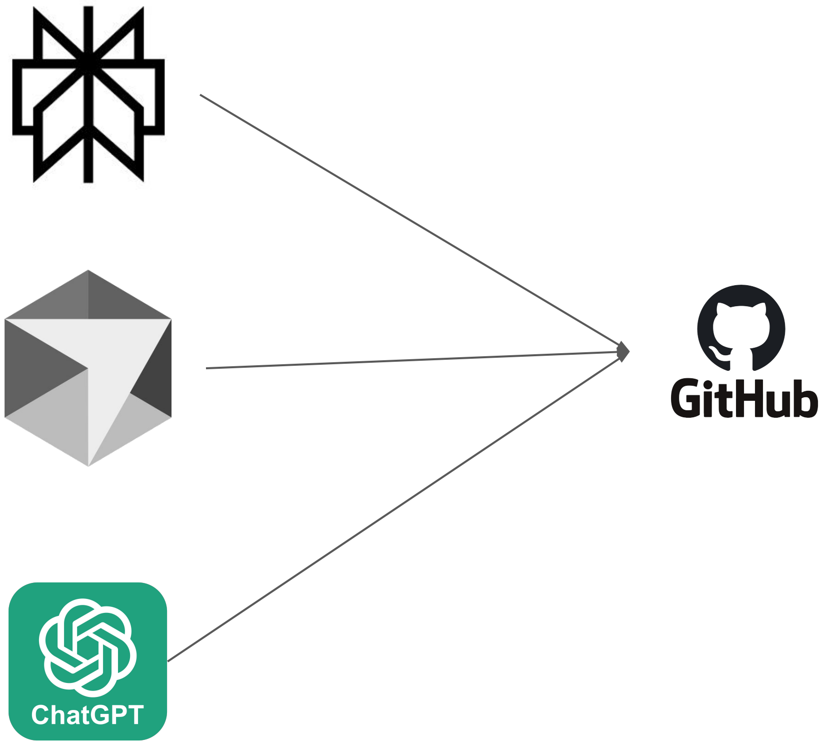
Problem with Tools

Maintenance Problem

Security fragmentation

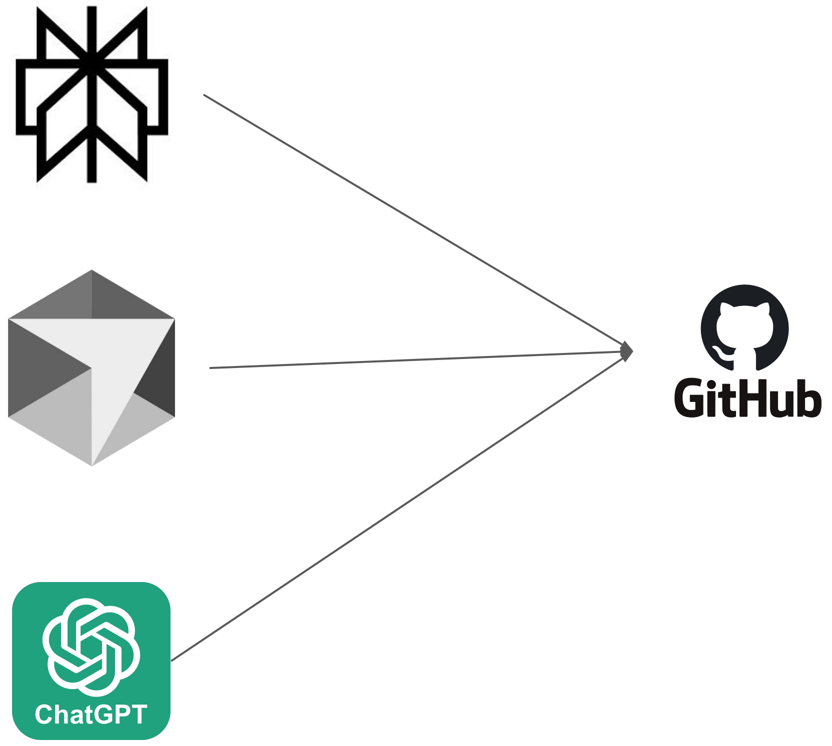
Cost and time wastage

Overview of the Problem



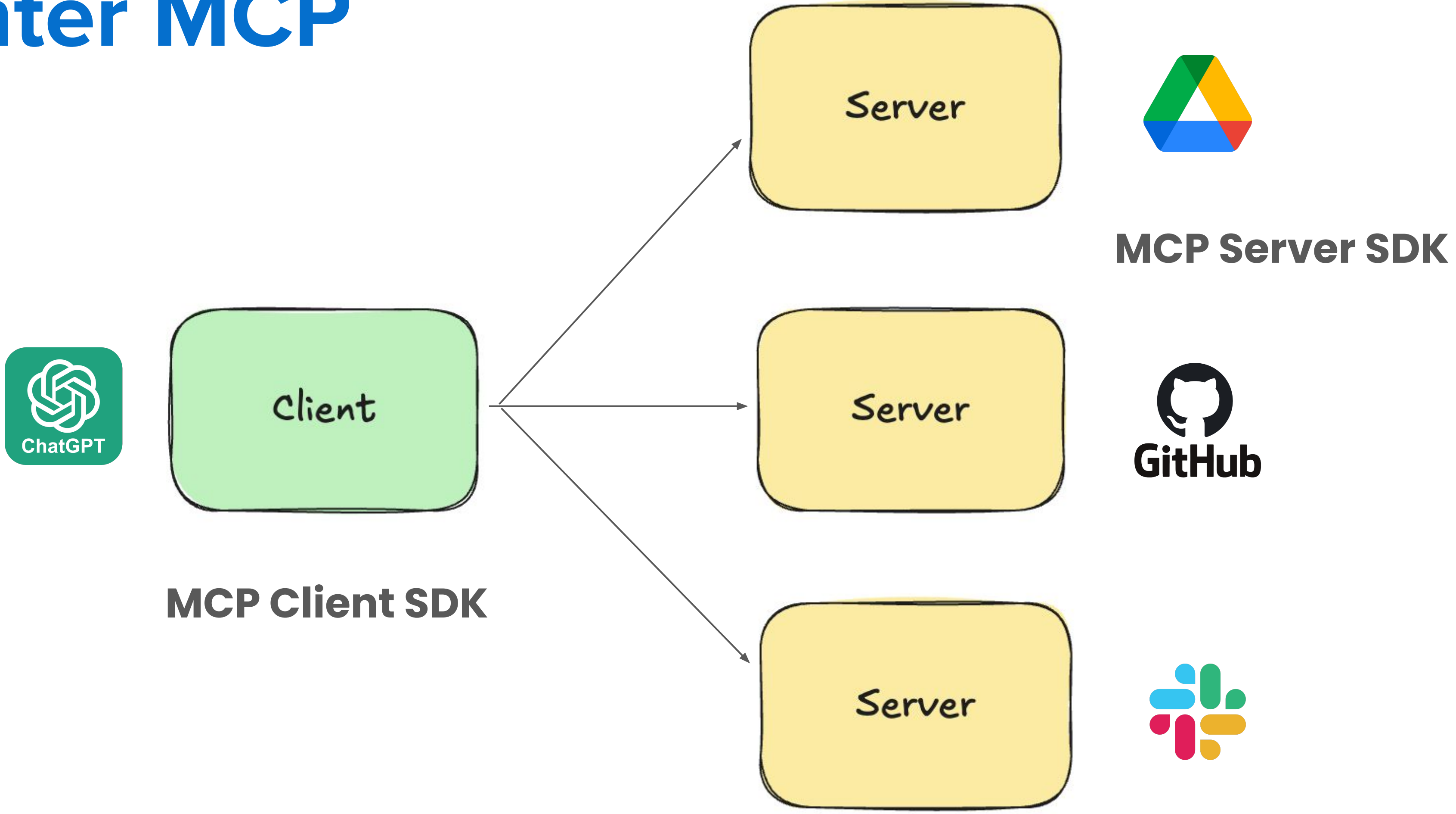
Every AI tool was building its own way to call every API.

The Solution



Github builds an integration that can be used by any AI Tool

Enter MCP


Server
MCP Server SDK
ChatGPT
Client
Server
GitHub
MCP Client SDK
Server

MCP vs Function/Tool Calling

```python
def get_weather(city: str):
    url = f"https://myweatherapi.com/weather?city={city}&key=API_KEY"
    response = http_get(url)
    data = json_parse(response)
    return f"{city}: {data['temp_c']}°C, {data['condition']}"
```

Client

```python
@app.route("/weather")
def weather_endpoint():
    city = request.args.get("city")
    # Query internal weather database
    result = lookup_weather(city)    #
    return jsonify(result)
```

Server

MCP vs API

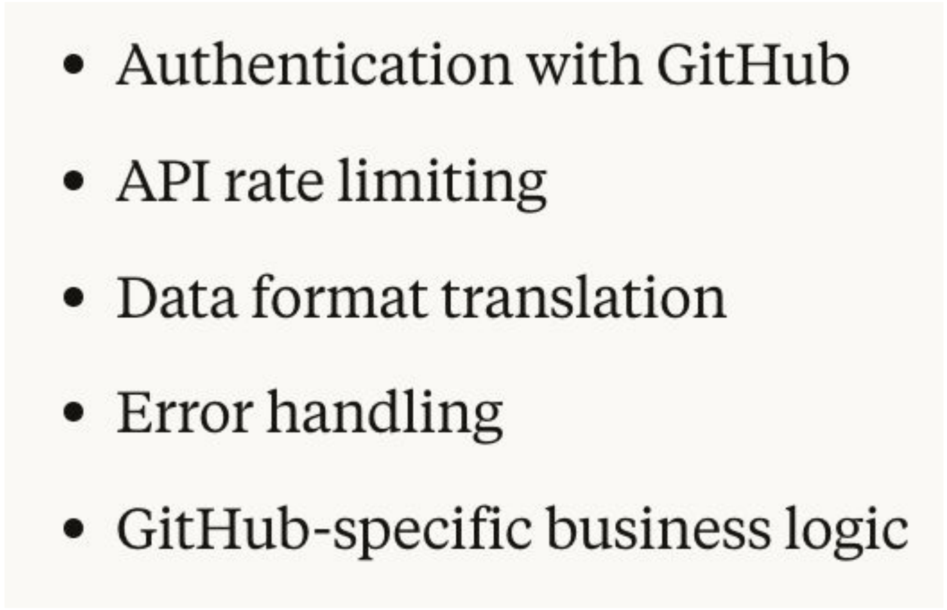
Server

```python
from mcp.server import Server

server = Server("WeatherServer")

@server.tool("get_weather")
def get_weather(city: str):
    data = lookup_weather(city)  # e.g.,
    return {
        "temperature": data["temp_c"],
        "condition": data["condition"]
    }

server.start()
```

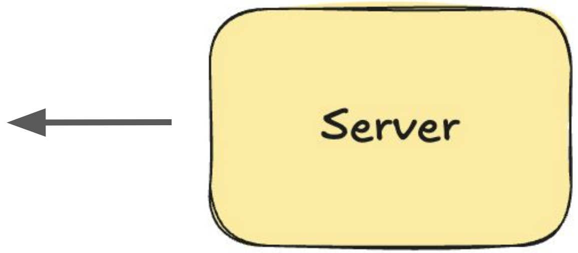
Client

```python
# AI client just calls tools via MCP
result = call_tool("get_weather", {"city": "London"})
```

Server does the Heavy Lifting



- Authentication with GitHub
- API rate limiting
- Data format translation
- Error handling
- GitHub-specific business logic



The Client has to just connect to the server using the same language

Benefits

N clients and M servers = M + N integrations

No maintenance overhead

Reduced cost and time

Better security

MCP Ecosystem

More AI Chatbots supporting MCP → More valuable for services to build MCP servers

More MCP servers available → More valuable for AI tools to support MCP

More adoption → More standardization → More ecosystem value

Not supporting MCP meant being cut off from the rapidly growing ecosystem