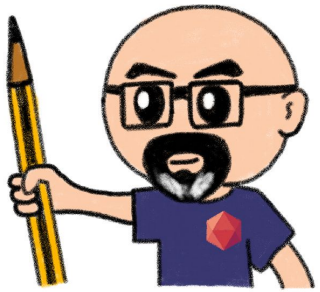
CONFERENCE MADRID 2025

Introducción al MCP: Conectando LLMs con tus Aplicaciones y Datos...

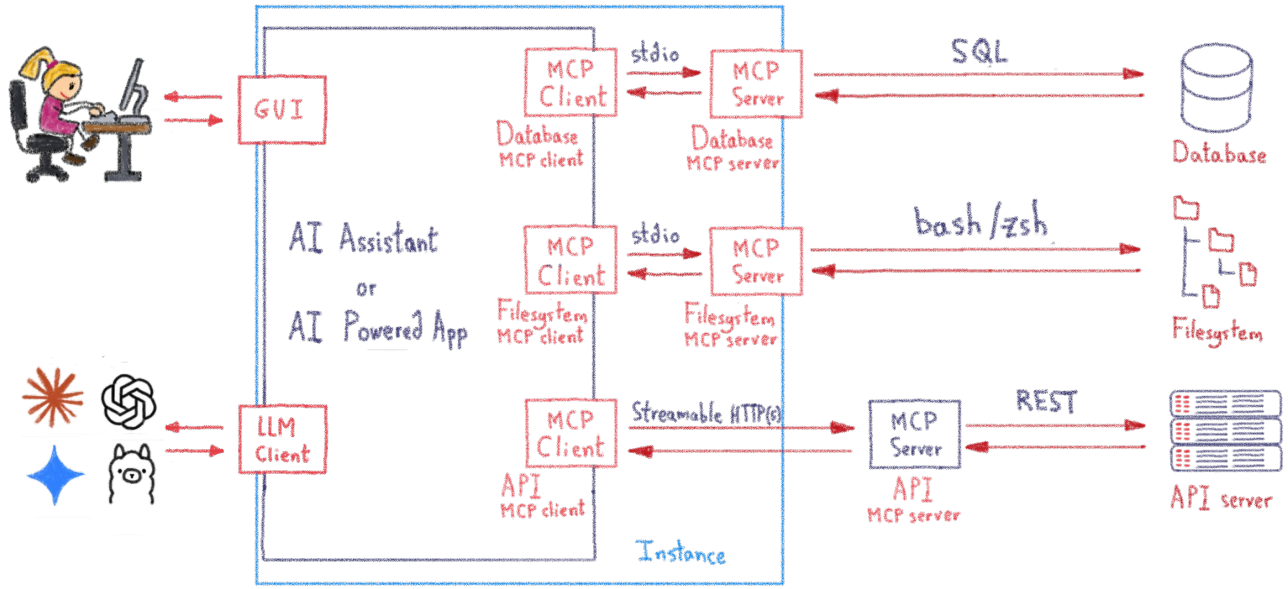
Horacio Gonzalez

2025-04-21



@LostInBrittany

clever cloud



Horacio Gonzalez

CONFERENCE MADRID 2025

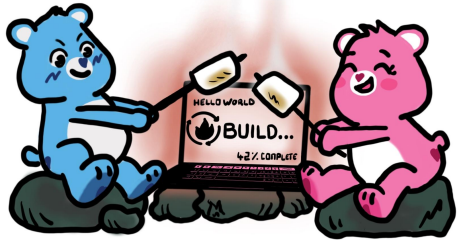
@LostInBrittany

Spaniard Lost in Brittany

Head of DevRel

clever cloud

Finist Devs



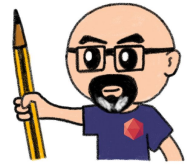
@LostInBrittany

Clever Cloud

clever cloud

From Code to Product

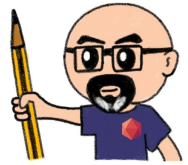
Our mission: give more **speed** to your **teams**
and better **quality** to your **projects**



@LostInBrittany

Summary

1. Introduction
2. LLM evolution
3. Model Context Protocol (MCP)
4. Architecture of MCP
5. MCPs are APIs
6. Q&A and discussion

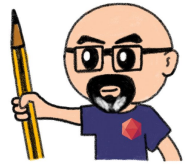


Introduction

LLMs are changing software development, they say...
how about you?

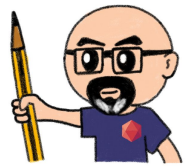
Why are we talking about this?

LLMs are changing development,
but individual devs don't always leverage them



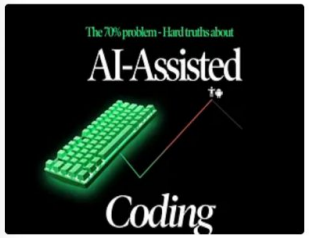
How do you use LLMs for your dev job?

1. Who here has already used LLM?
2. Who here has already used LLM professionally?
3. Who here has already used LLM to assist with code?
4. Who here has already used LLMs by coding?



How LLMs are changing dev jobs

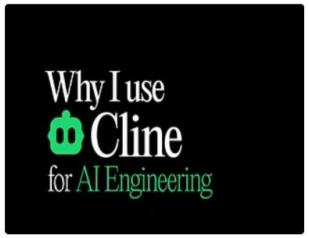
Featured AI articles



The 70% Problem in AI Coding

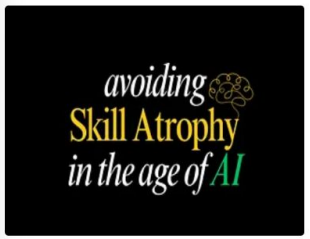
Maximizing the Human 30%

MCP: The Model Context Protocol

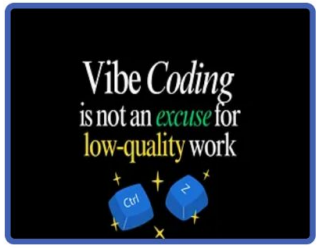


Why I Use Cline for AI Engineering

Leading Engineering Teams in the GenAI Era

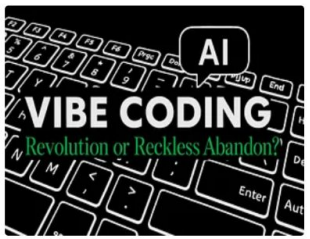


Avoiding Skill Atrophy in the Age of AI



Vibe Coding is not an excuse for low-quality work

What are AI Agents? why do they matter?

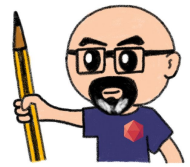


Vibe Coding: Revolution or Reckless Abandon?

AI-Driven Prototyping: v0, Bolt, and Lovable Compareda

A point of view I find balanced: Addy Osmani

https://addyosmani.com/

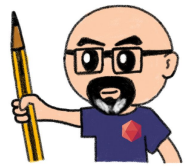


LLMs come in different flavors

Not all LLMs are created equal

They have different trade-offs
in capabilities, accessibility, and control

Choosing the right one depends on your
use case, security needs, and infrastructure



@LostInBrittany

Closed-source LLMs (Cloud-based APIs)

📌 Examples

- OpenAI (ChatGPT), Anthropic (Claude), Google (Gemini), Microsoft (Copilot)

✅ Advantages:

- Powerful and well-trained (best models available)
- Easy to use via APIs
- Regularly updated & improved

❌ Challenges:

- Black box (you don't control how they work)
- Expensive (API calls can add up quickly)
- Data privacy concerns (sending requests to external servers)

💡 When to use?

- If you need the most advanced models and don't mind API costs or external dependencies

@LostInBrittany

clever cloud

Open-source LLMs (Self- or cloud-hosted)

📌 Examples

- Meta's Llama 3, Mistral, Google's Gemma, Alibaba's Qwen

✅ Advantages:

- Greater control (you know exactly how the model works)
- Can be fine-tuned for specific needs
- No external API costs

❌ Challenges:

- Requires more setup (you have to run the model yourself)
- May not be as powerful as the latest closed models
- Needs infrastructure (e.g., GPUs for hosting)

💡 When to use?

- If you need control over the model & lower costs but are okay with slightly weaker performance

clever cloud

Local models (on your machine or server)

📌 Examples

- Ollama, GGUF-based models (e.g., Llama, Mistral, Mixtral)

✅ Advantages:

- Works offline (great for security-sensitive applications)
- No API costs (completely free to use once set up)
- Low latency (responses are instant if hardware is good)

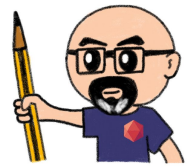
❌ Challenges:

- Limited by your hardware (needs a strong CPU/GPU)
- Not always as capable as cloud-hosted models
- Setup complexity (installing and optimizing models)

💡 When to use?

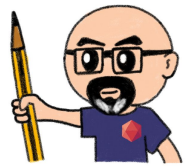
- If you need privacy and control, and you have the hardware to run an LLM efficiently

clever cloud



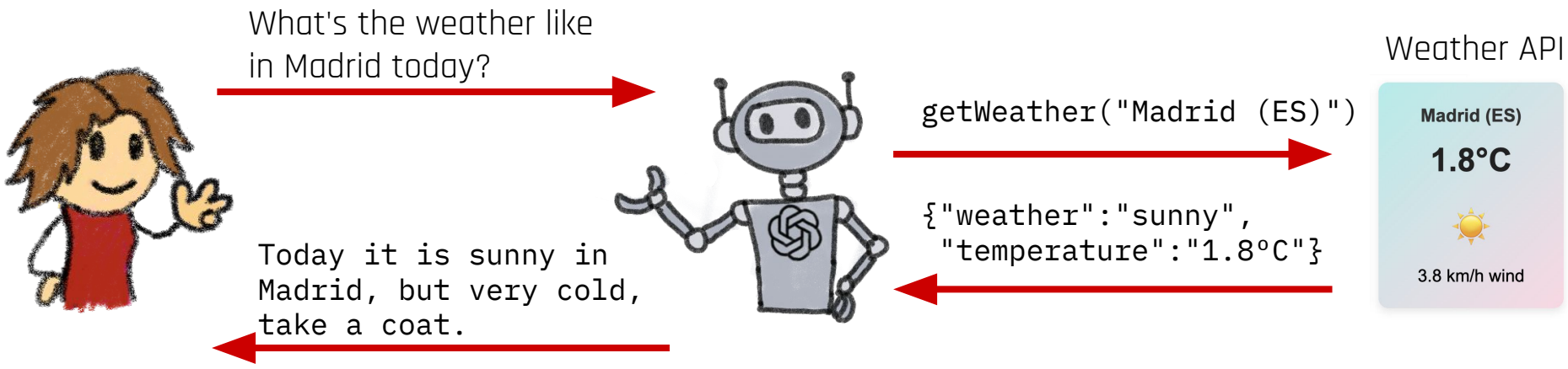
Choosing the Right Model for your Apps

- Cloud APIs
  - Great for rapid development, but costly and not always secure
- Self-hosted open models
  - Best balance for long-term control and scalability
- Local models
  - Best for privacy-sensitive applications



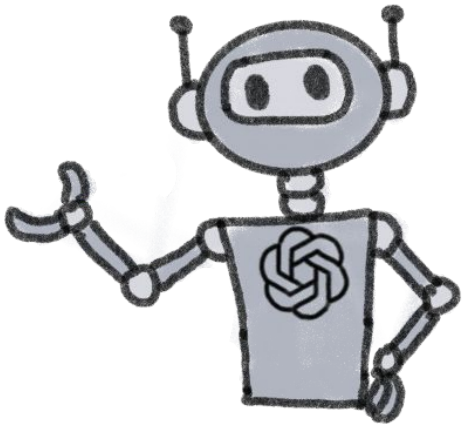
LLM evolution

From simple chat to tool-enhanced agent!



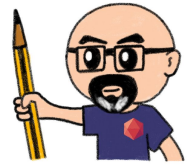
LLM are only language models

What's the weather like in Madrid today?



```
I'm unable to provide real-time
information or current weather updates.
```

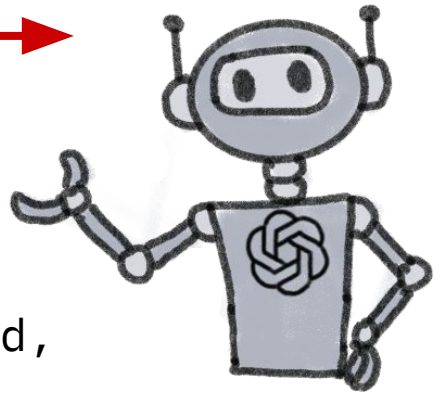
They have no built-in way to use external tools or real-time data



@LostInBrittany

Tools and plugins were added

What's the weather like in Madrid today?



`getWeather("Madrid (ES)")`

Weather API

Madrid (ES)

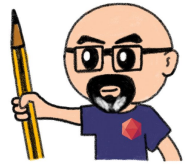
1.8°C

3.8 km/h wind

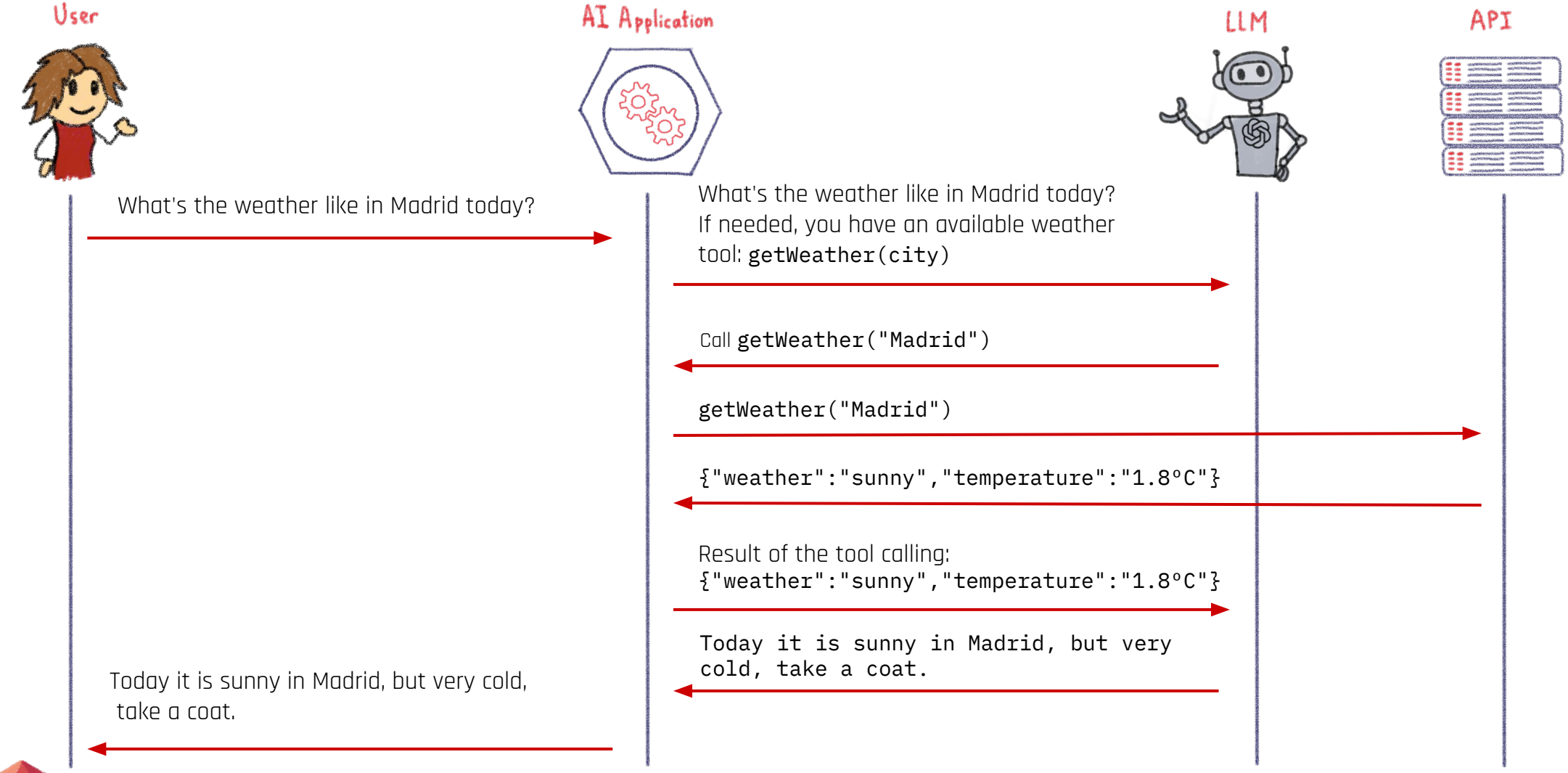
`{"weather":"sunny", "temperature":"1.8°C"}`

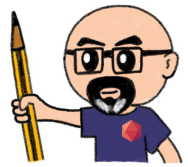
`Today it is sunny in Madrid, but very cold, take a coat.`

LLM recognizes it needs an external function and calls it, integrating the result into a natural-language response.



LLM don't call directly those tools





@LostInBrittany

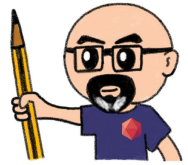
clever cloud

How are those LLM Tools defined?

LyingWeatherTool.java

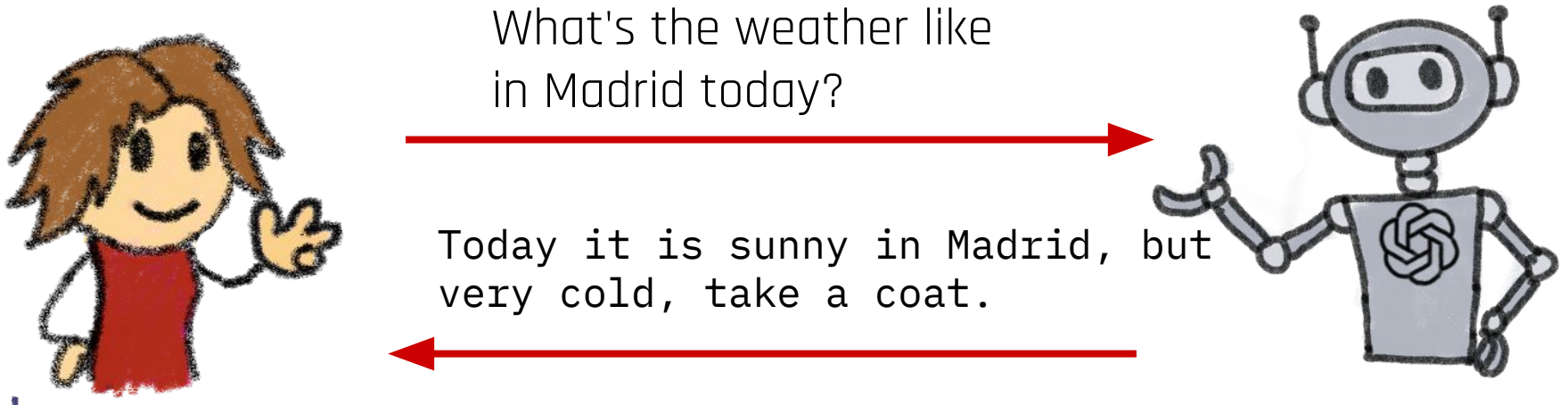
```java
//DEPS dev.langchain4j:langchain4j:1.0.0-beta1

import dev.langchain4j.agent.tool.Tool;

public class LyingWeatherTool{
 @Tool("A tool to get the current weather in a city")
 public static String getWeather(String city) {
   return "The weather in " + city + " is sunny and hot.";
 }
}
```

Here in Java using LangChain4j

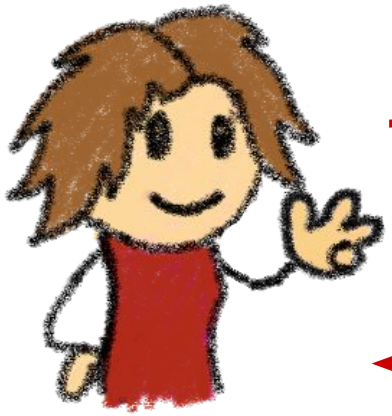


Why this matters?

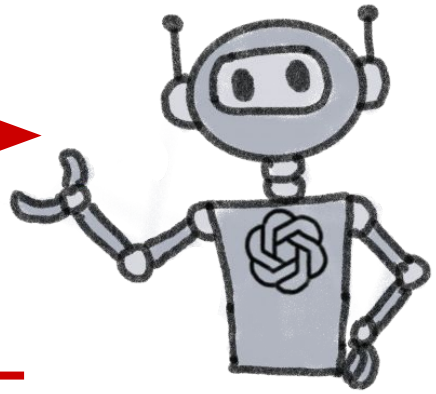
- Moves LLMs from static text generation
  - dynamic system components
- Increases accuracy & real-world usability
- Allows developers to control what the LLM can access



From LLM chats to LLM-powered agents



Can you summarize this YouTube video?



YouTube

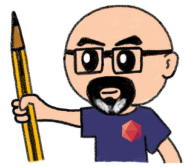
`Of course,the video is a talk of Horacio about MCP…`

Whisper

*This is a "fake" view, remember LLMs dont call tools directly
But it's the view from the Point of View of the user

LLMs act like an agent that can plan actions: search the web, run some code, then answer

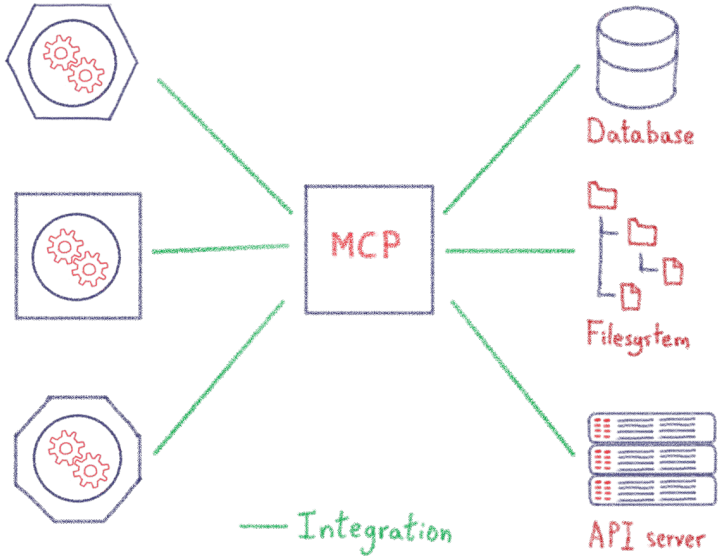
clever cloud



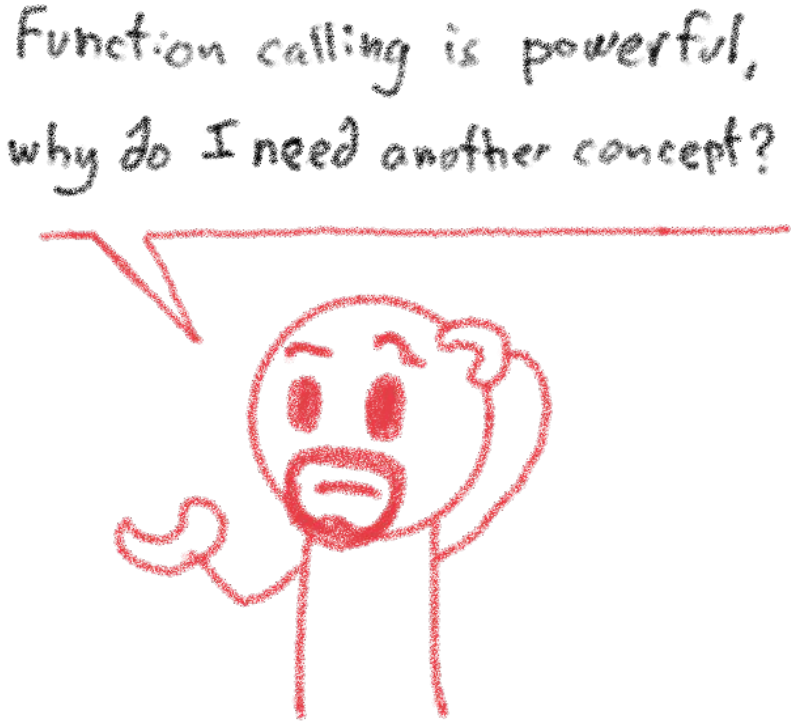
@LostInBrittany

Model Context Protocol (MCP): The missing link

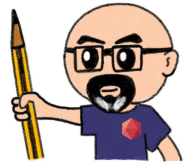
MCP bridges LLMs with your applications, enabling controlled, real-world interactions



Why Do We Need MCP?



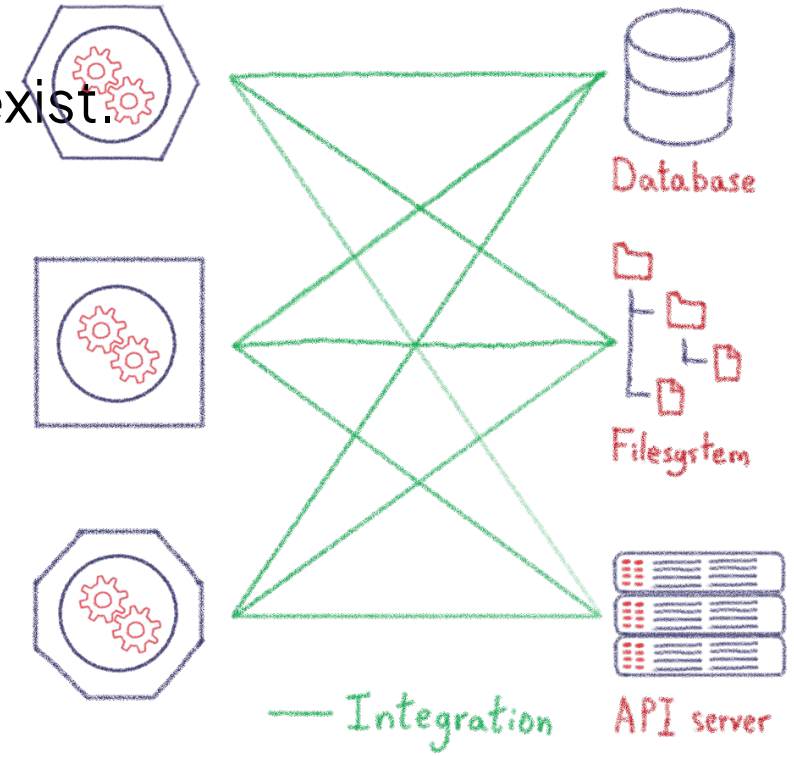
LLM function calling is useful,
but it lacks structure

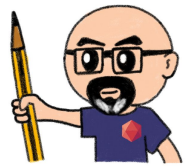


Why Do We Need MCP?

Problem

- LLMs **don’t automatically know** what functions exist.
- **No standard way** to expose an application's capabilities.
- **Hard to control** security and execution flow.
- Expensive and fragile **integration spaghetti**



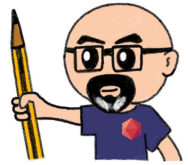


@LostInBrittany

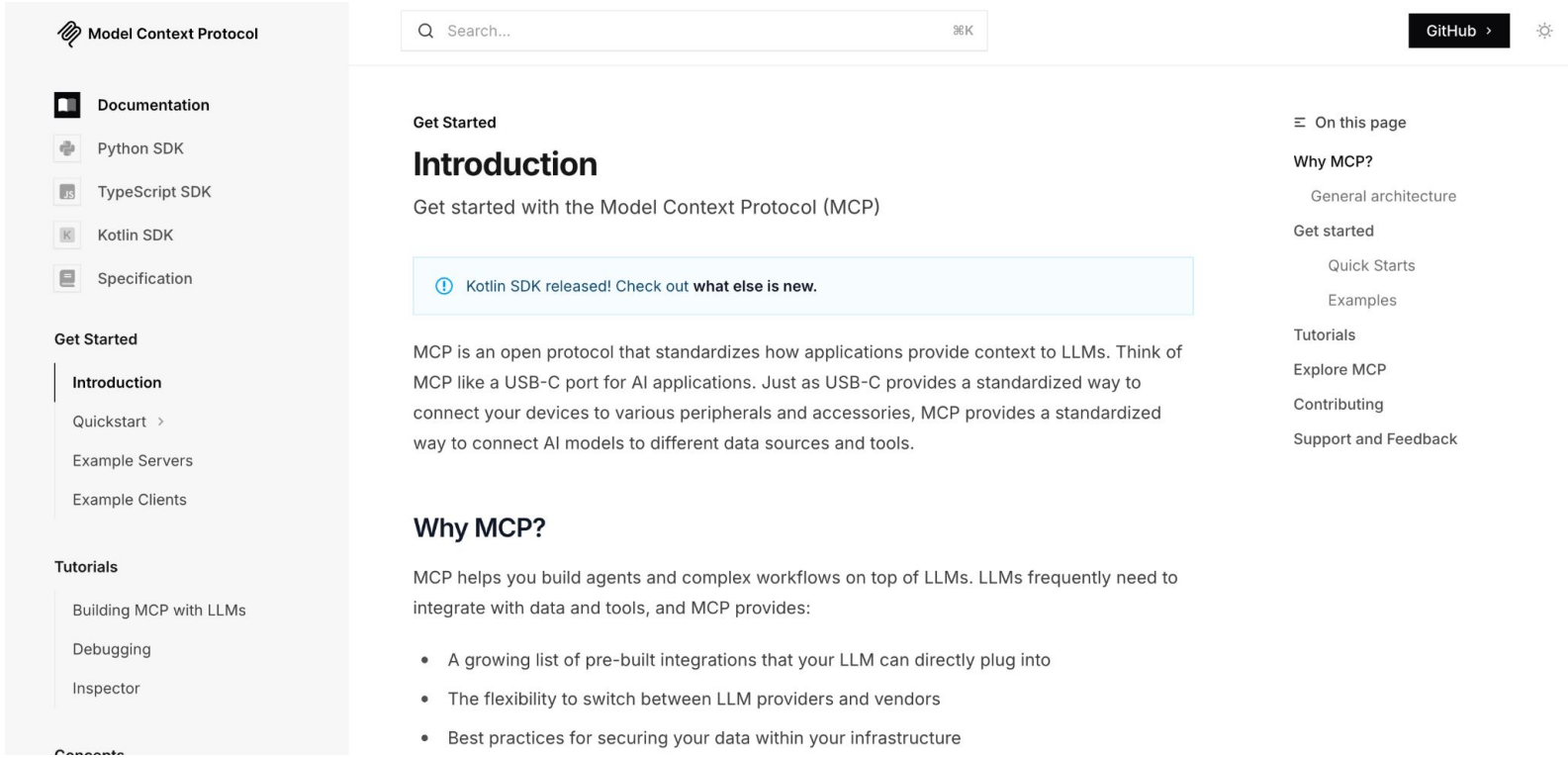
clever cloud

Model Context Protocol

Anthropic, November 2024:
LLMs intelligence isn't the bottleneck, connectivity is



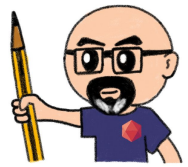
Model Context Protocol



De facto standard for exposing system capabilities to LLMs

https://modelcontextprotocol.io/

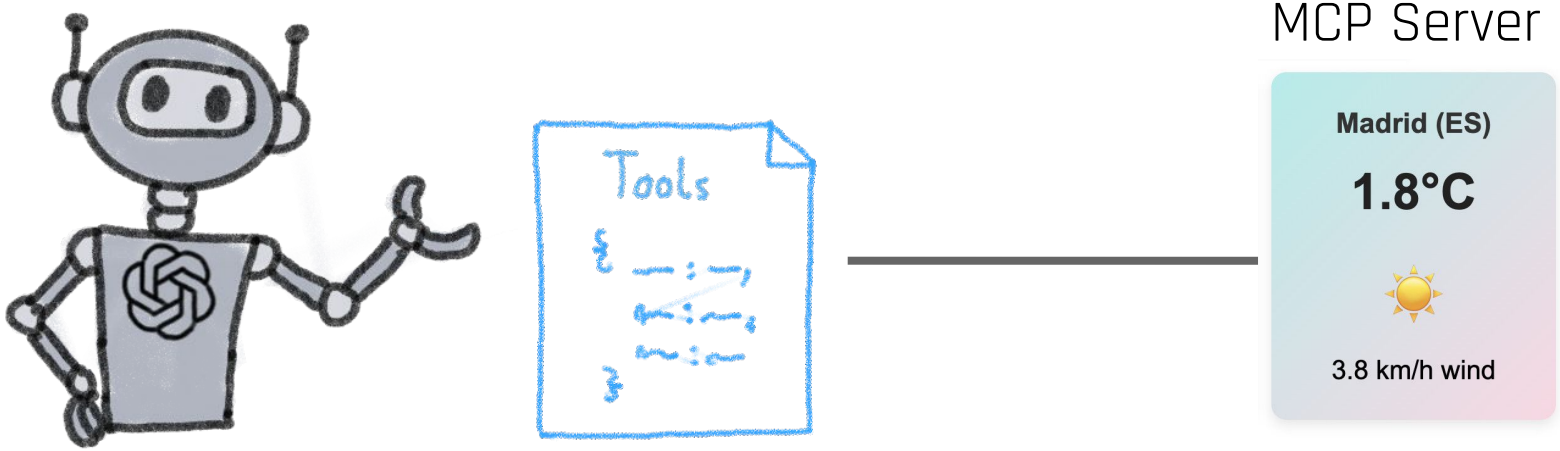
clever cloud



@LostInBrittany

How MCP works

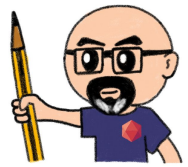
- Applications define an MCP manifest (structured JSON).
- The manifest describes available functions, input/output formats, and security policies.
- LLMs can discover and request function execution safely.



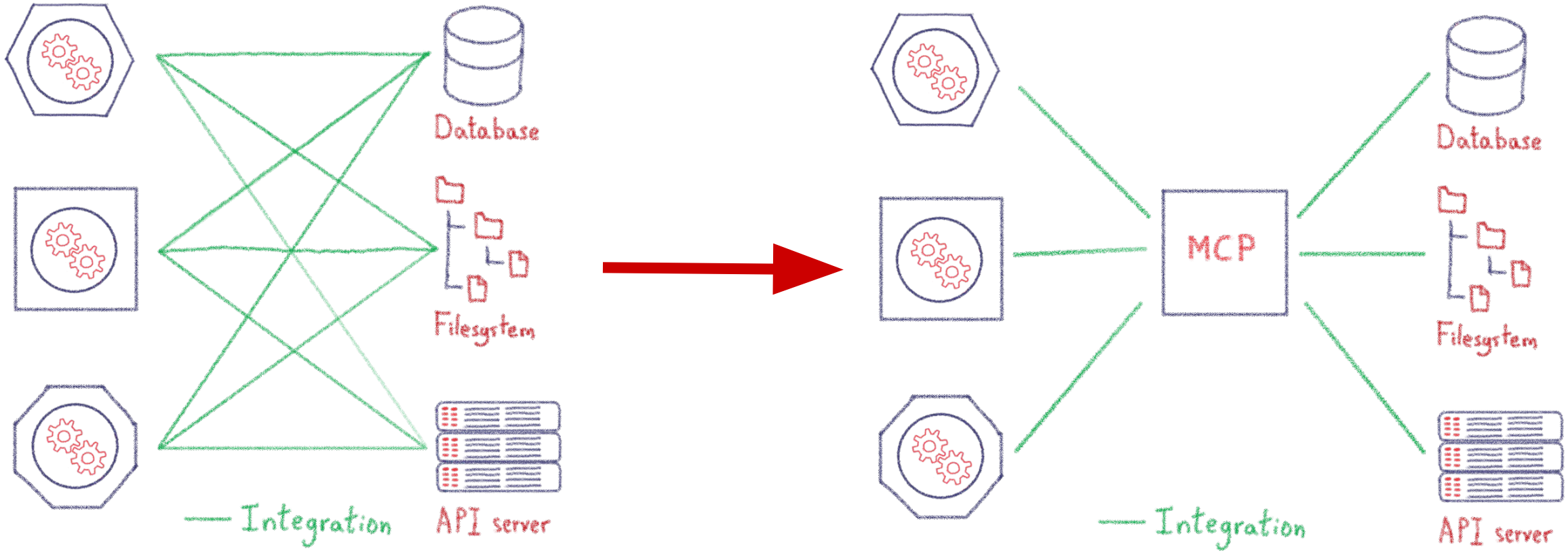
MCP is provider-agnostic

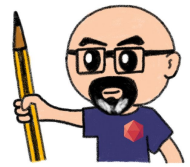
Works with any LLM provider

Ensures standardized function exposure across platforms



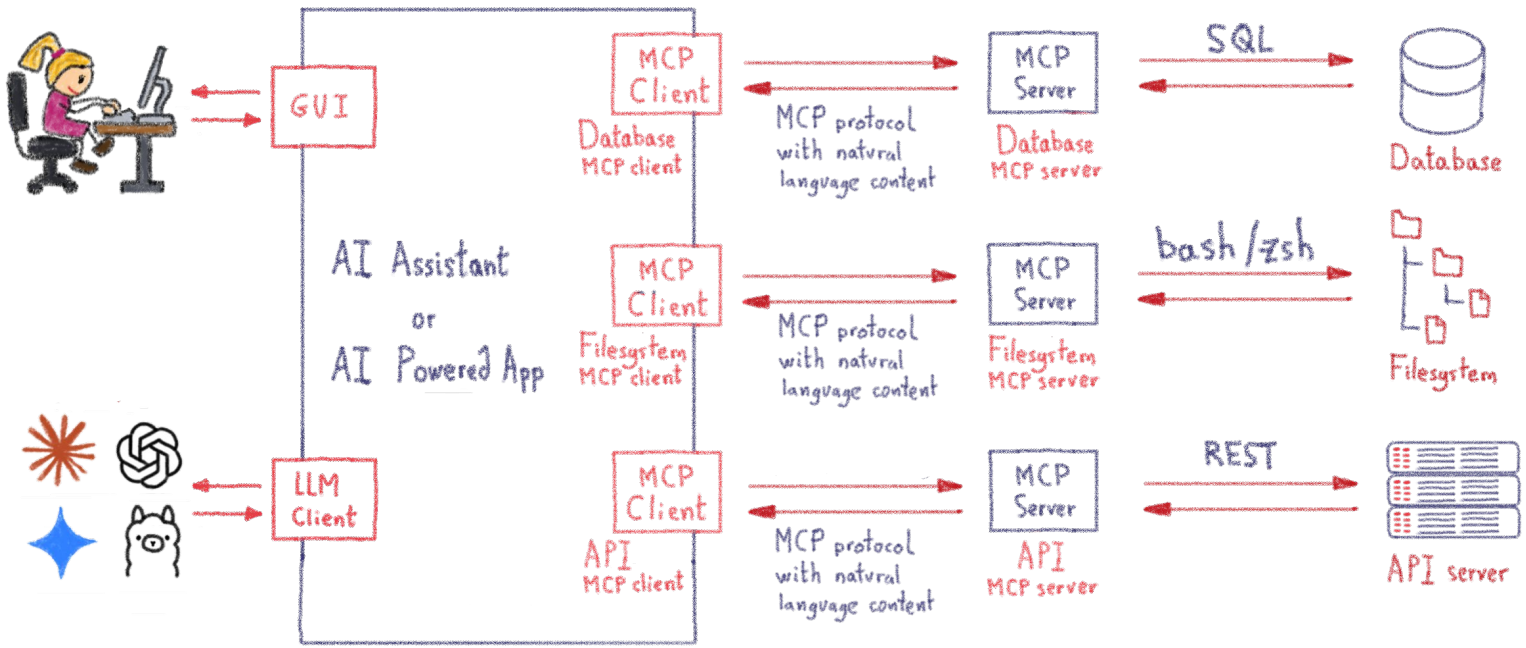
MCP solves integration spaghetti



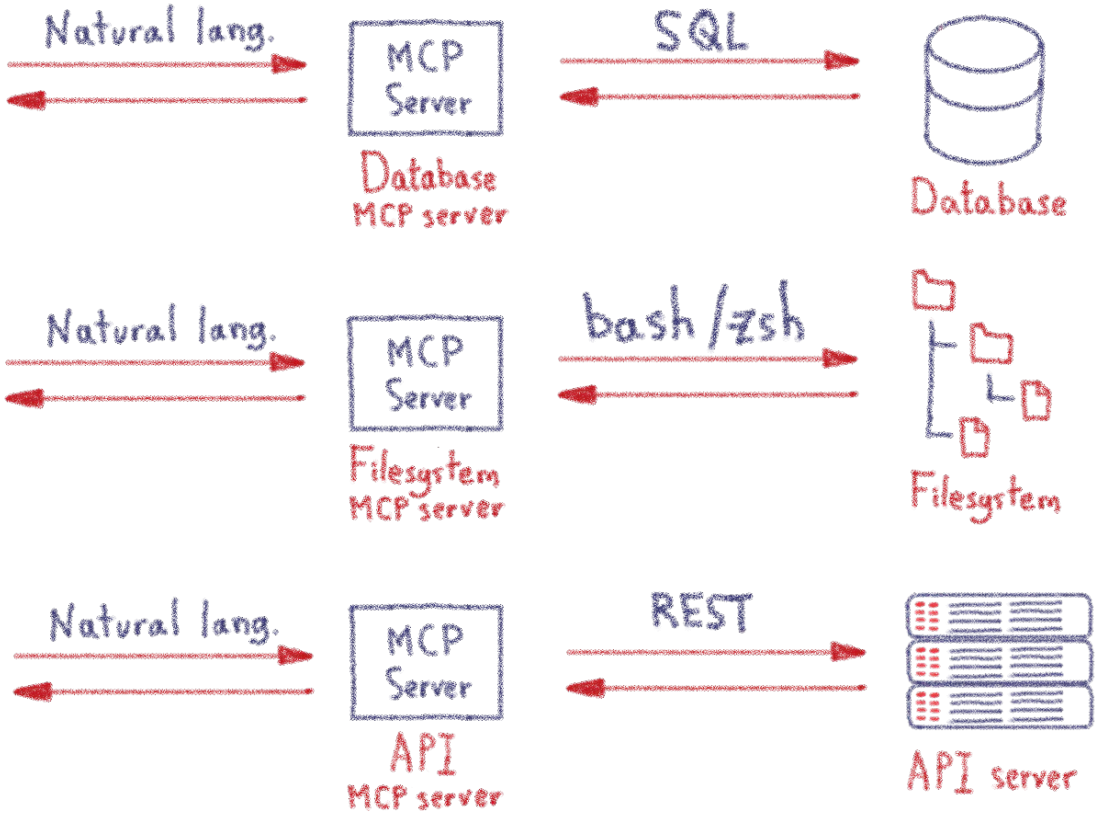


The architecture of MCP

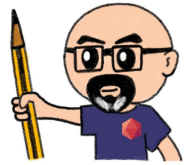
Clients, servers, protocol and transports
Tools, resources and prompts



MCP Servers: APIs in natural language



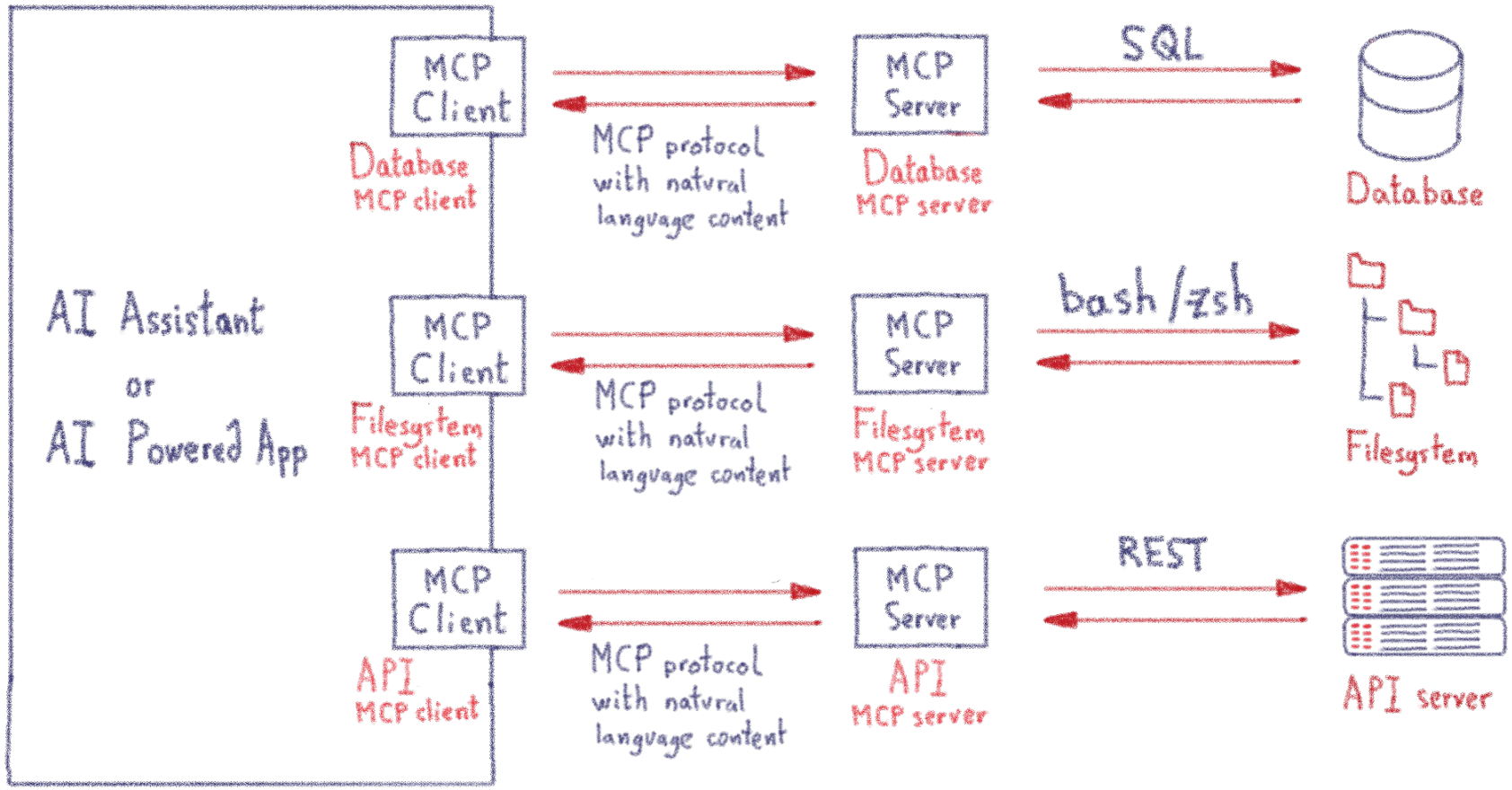
A new kind of API



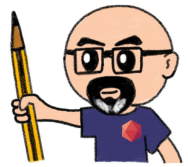
@LostInBrittany

clever cloud

MCP Clients: on the AI assistant or app side



One MCP client per MCP Server



@LostInBrittany

clever cloud

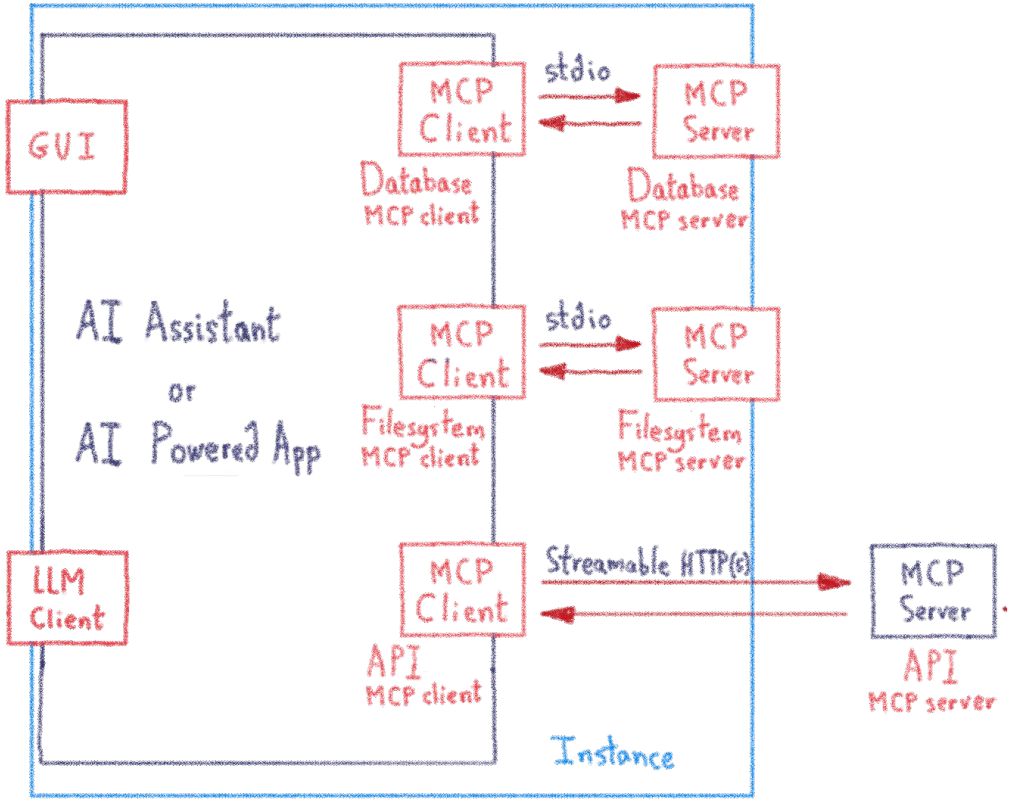
MCP Protocol & Transports

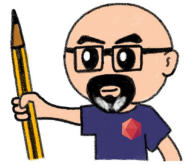
MCP Protocol

Follow the JSON-RPC 2.0 specification

MCP Transports

- STDIO (standard I/O)
  - Client and server in the same instance
- HTTP with SSE transport (deprecated)
- Streamable HTTP
  - Servers SHOULD implement proper authentication for all connections

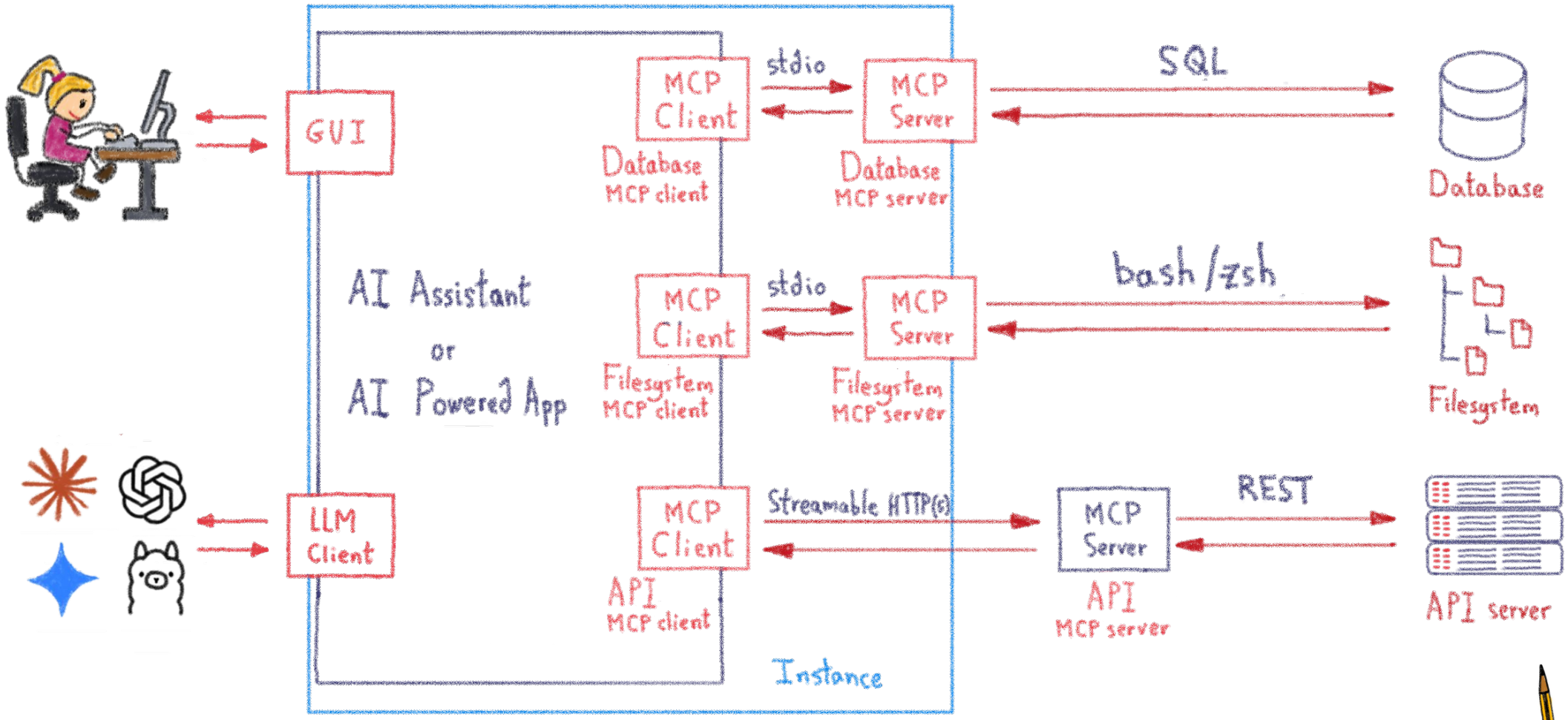


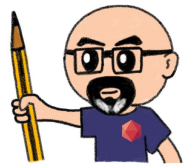


@LostInBrittany

clever cloud

Full MCP architecture



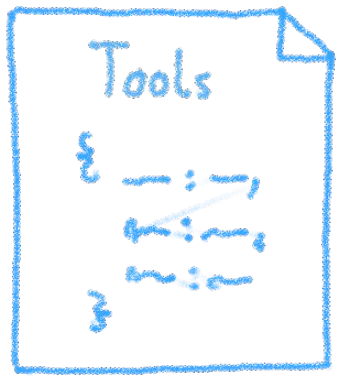


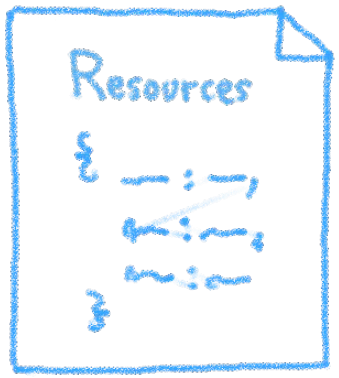
@LostInBrittany

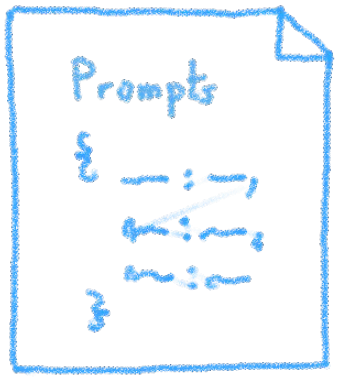
clever cloud

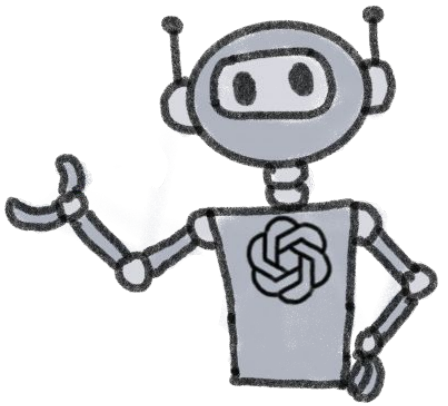
Services: tools, resources & prompts

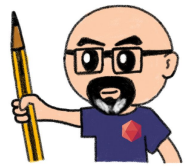
- Tools
  - Standardized way to expose functions that can be invoked by clients
- Resources
  - Standardized way to expose resources to clients
  - Each resource is uniquely identified by a URI
- Prompts
  - Standardized way to expose prompt templates to clients
  - Structured messages and instructions for interacting with LLMs







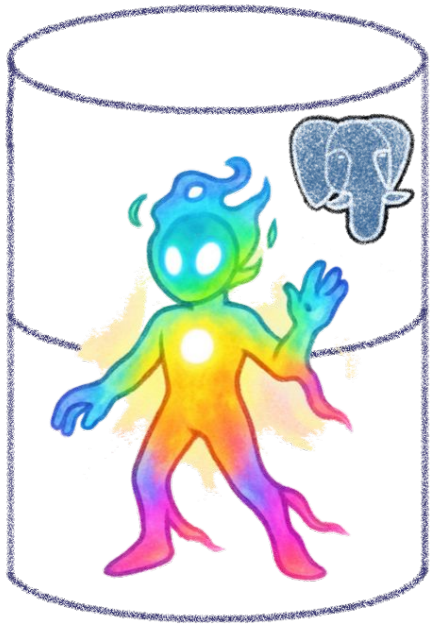




@LostInBrittany

MCPs are APIs

And they should be architectured in a similar way



Let's use an example: RAGmonsters

README License

🧩 RAGmonsters Dataset

Overview

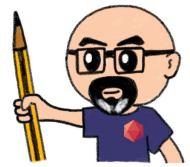
The RAGmonsters dataset is a collection of 30 fictional monsters created specifically for demonstrating and testing Retrieval-Augmented Generation (RAG) systems. Each monster is completely fictional and contains detailed information that would not be found in an LLM's training data, making it perfect for showcasing how RAG can enhance an LLM's knowledge with external information.

Purpose

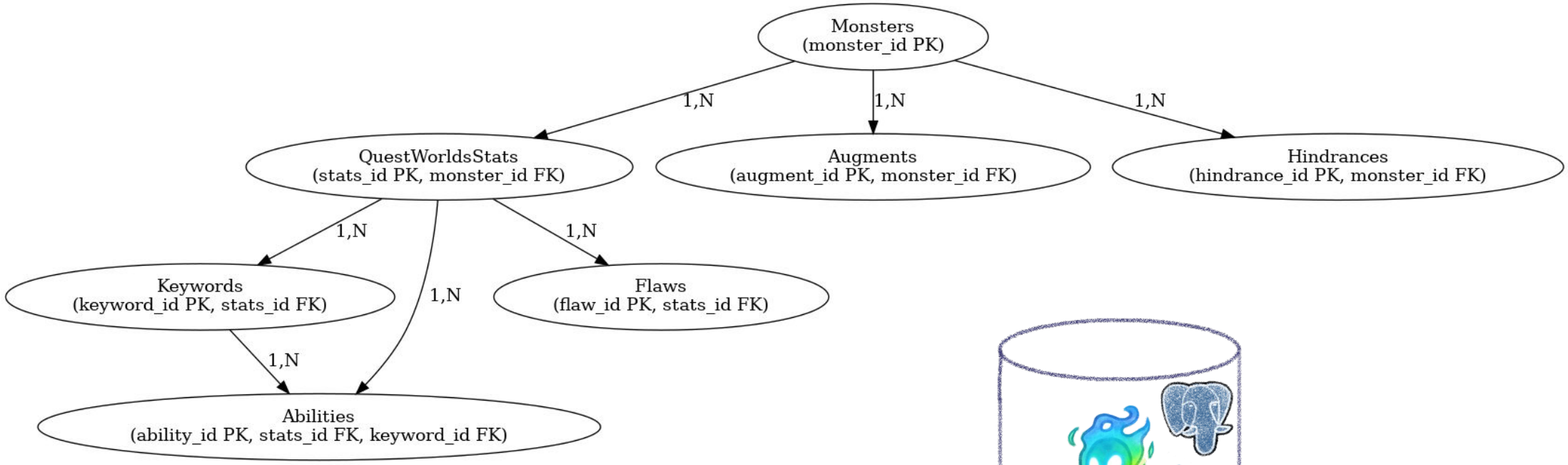
This dataset serves several educational purposes:

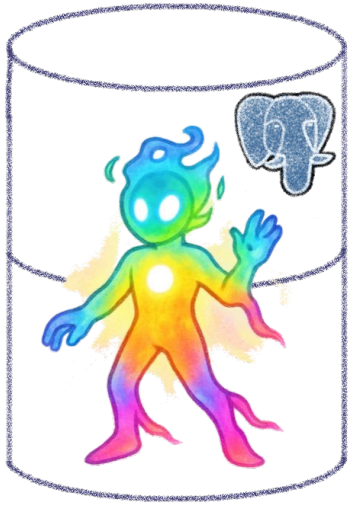
1. **Demonstrates RAG Value**: Shows how RAG can provide accurate answers about topics not in the LLM's training data
2. **Tests Retrieval Quality**: The varied attributes and relationships allow testing of different retrieval methods
3. **Supports Advanced Features**: Perfect for demonstrating filtering, re-ranking, and hybrid search techniques
4. **Provides Engaging Content**: Makes learning RAG concepts more fun and memorable

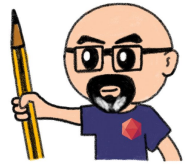
https://github.com/LostInBrittany/RAGmonsters



RAGmonsters PostgreSQL Database







@LostInBrittany

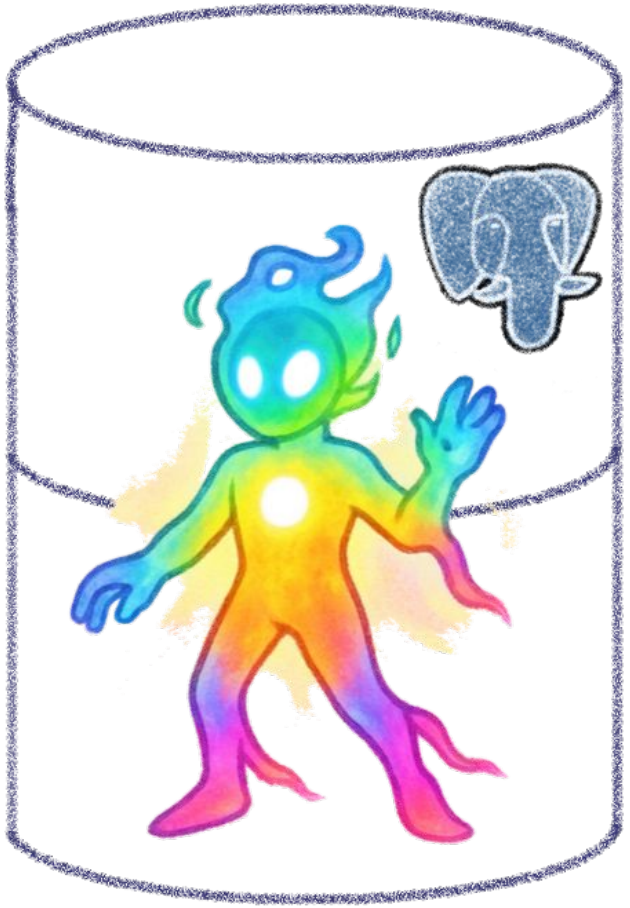
clever cloud

We want to allow LLM request it

Two options:

- A generic PostgreSQL MCP server
- A custom-made MCP server tailored for RAGmonsters

Which one to choose?



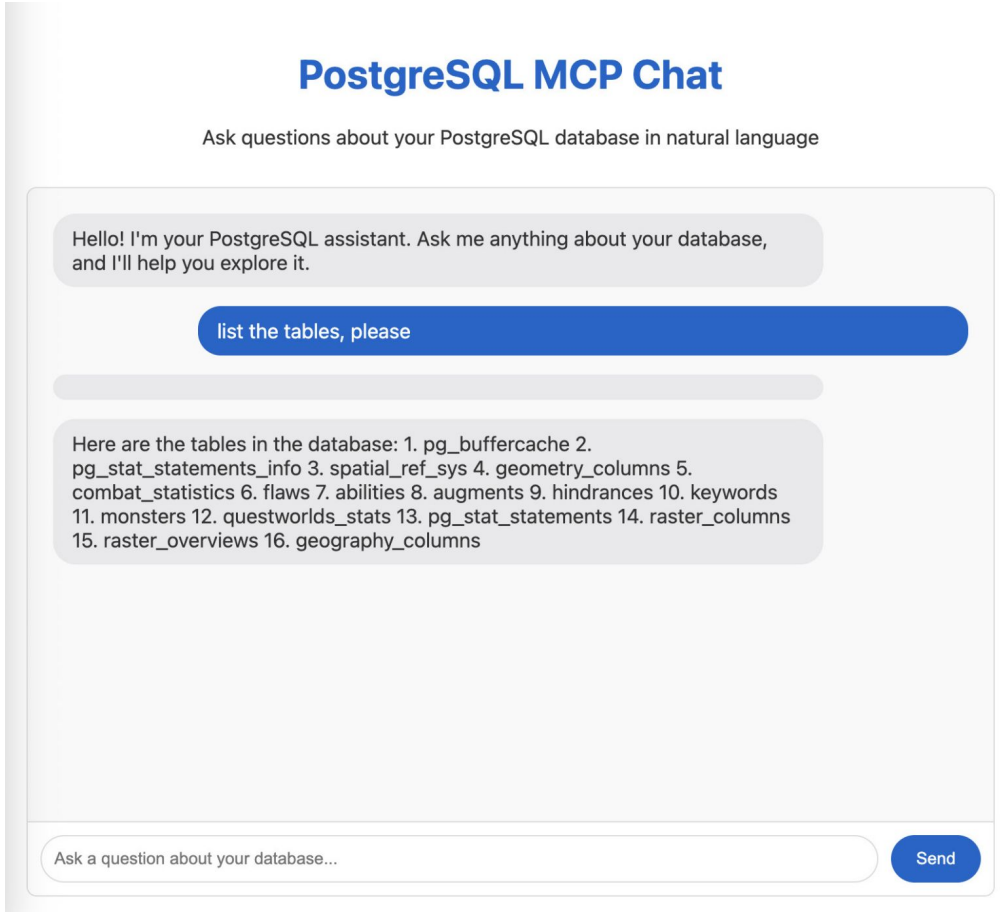
Generic PostgreSQL MCP server

Using PostgreSQL MCP Server

- A Resource that give the table schema for tables: `/schema`
- A Tool that allows to do SQL queries: `query`

LLM can know what tables do we have and what is their structure, and it can request them

Implementation:
https://github.com/CleverCloud/mcp-pg-example
PostgreSQL MCP Server:
https://github.com/modelcontextprotocol/servers/tree/main/src/postgres



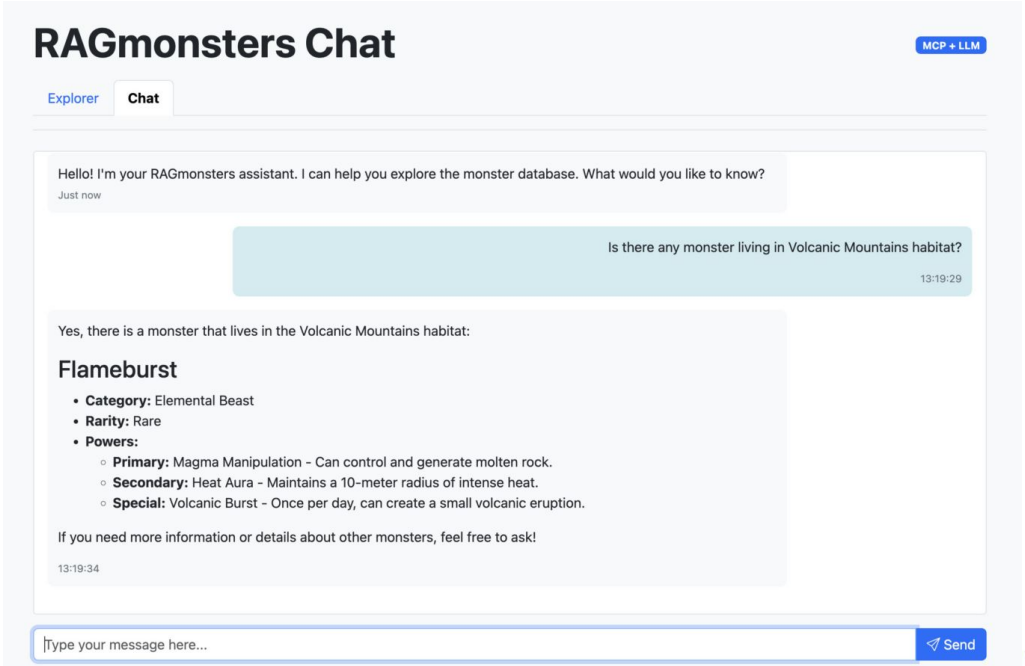
Custom-made RAGmonsters MCP server

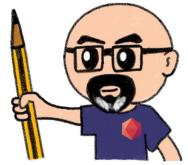
Coding a MCP server for it. It offers targeted tools:

- `getMonsterByName`: fetches detailed information about a monster.
- `listMonstersByType`: Lists monsters of a given type.

- Easy, intuitive interactions for LLMs.
- Optimized for specific use cases.
- Secure (no raw SQL).

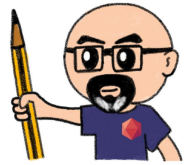
Implementation:
https://github.com/LostInBrittany/RAGmonsters-mcp-pg





How to choose?

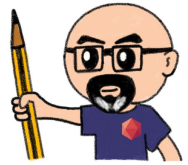
Aspect	Generic MCP Server	Domain-Specific MCP Server
Setup Speed	Fast, minimal configuration	Slower, requires planning
Efficiency	Lower, LLM must explore schema	High, optimized for specific tasks
Security	Risk of SQL injection	Secure, predefined tools
Flexibility	Adapts to any schema	Needs updates with schema changes
User Experience	Complex, LLM must learn	Simple, guided interactions



Conclusion

- Generic MCP servers: Quick to set up, flexible, but less efficient and more error-prone.
- Domain-specific MCP servers: Safer and faster for targeted tasks, but need more upfront design.
- Choose wisely: Use generic for exploration, domain-specific for production.

A bit like for REST APIs, isn't it?



@LostInBrittany

CONFERENCE
MADRID 2025

That's all, folks!

Thank you all!