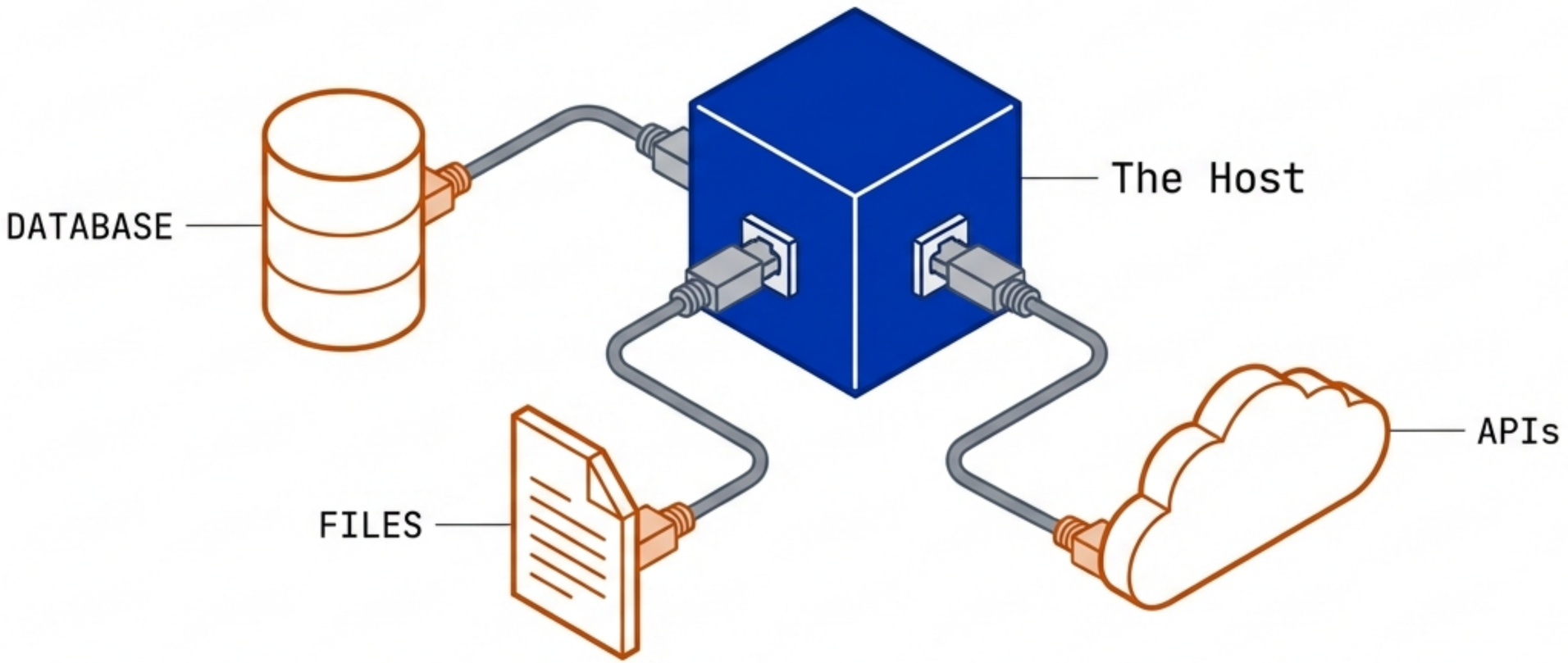
Model Context Protocol (MCP)

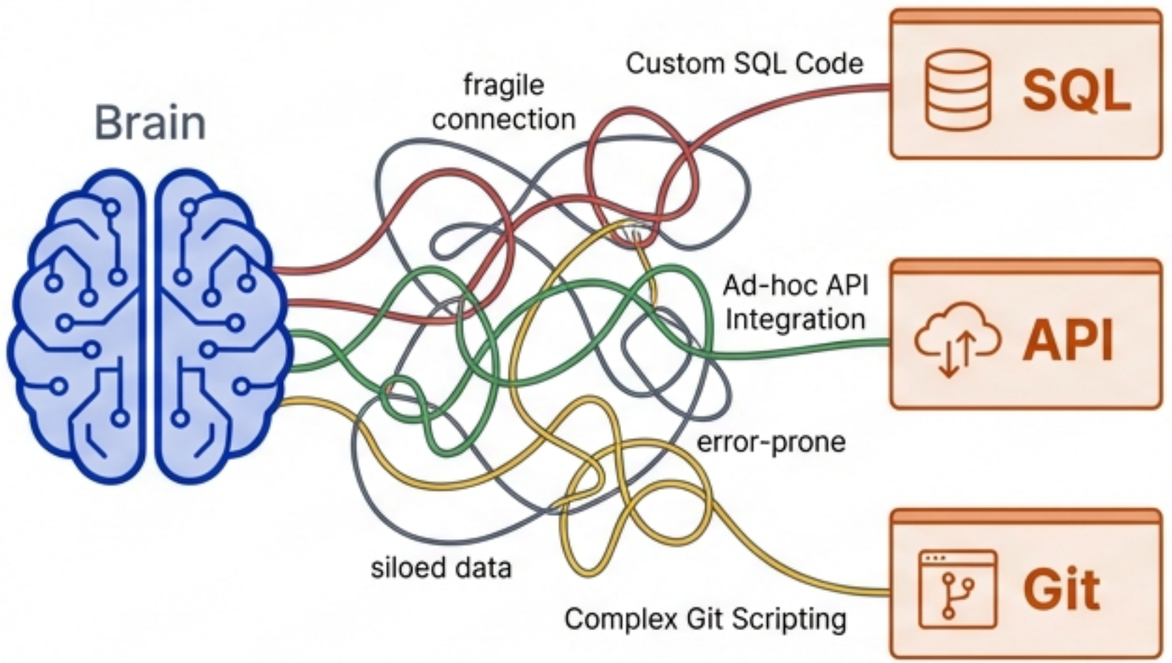
The Universal Standard for Connecting AI Models to Systems, Data, and Tools.



Created by Anthropic (2024) | Open Standard

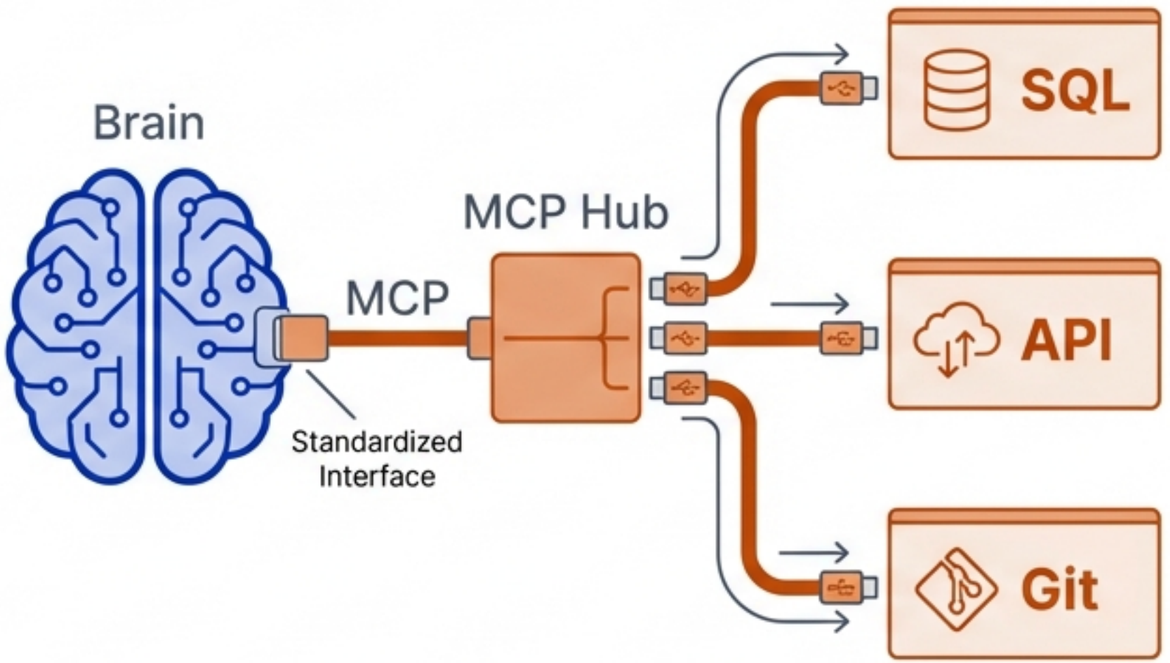
The ‘USB-C’ for AI Applications

The Old Way: Siloed Intelligence



Previously, connecting an LLM to data required bespoke integration code for every single data source. The systems were isolated and fragile.

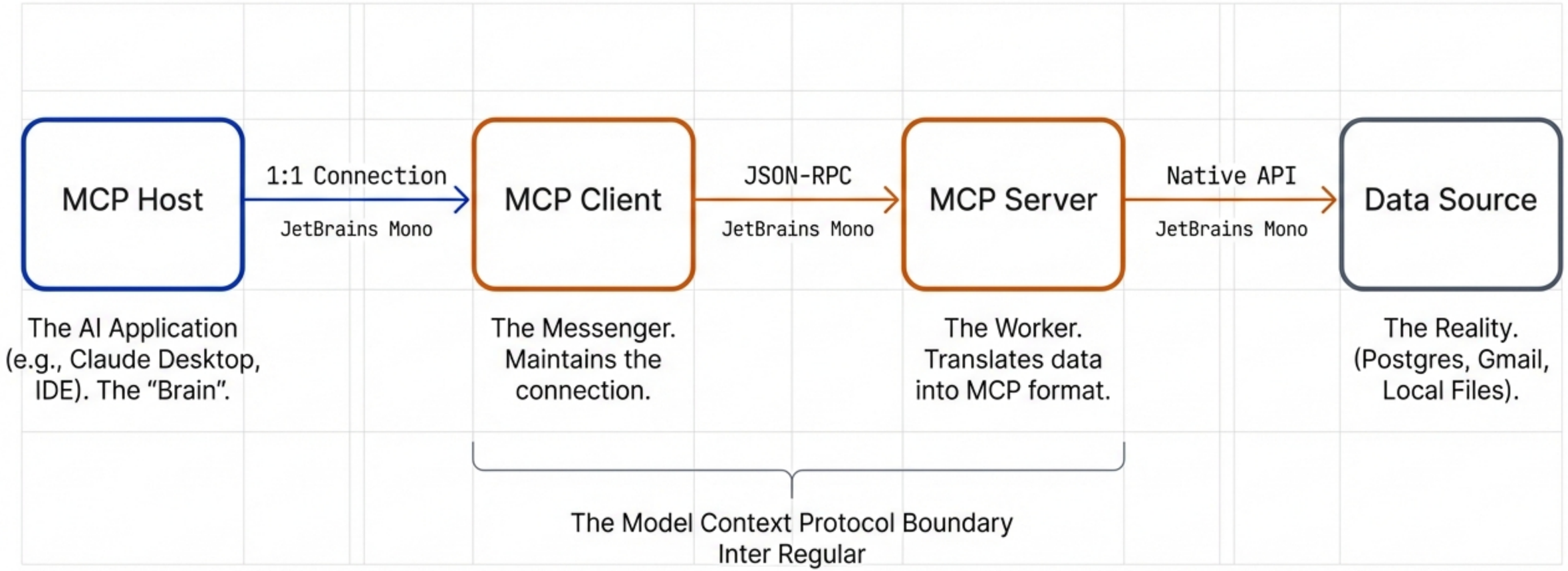
The MCP Way: Universal Connection



Just as a USB-C port accepts a hard drive, camera, or monitor without custom wiring, MCP lets AI plug into any data source via a standardized interface.

One Protocol, Thousands of Tools.

The Architecture of a Connection



The Three Primitives: How Servers Empower AI

1. Tools

Executable functions the AI can invoke to perform actions or side effects.

```
search_emails()
execute_sql_query()
```

2. Resources

Passive data the AI can read and analyze.

```
File contents
Database schemas
Logs
```

3. Prompts

Reusable templates to structure model behavior and interactions.

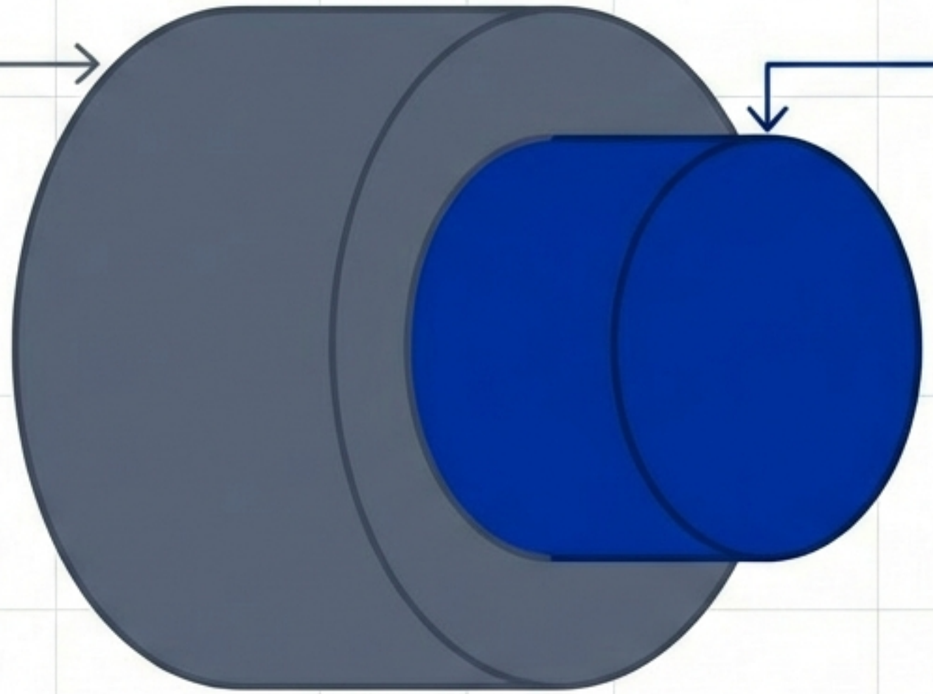
```
Bug Report Template
System Instructions
```

The Protocol Layers

Transport Layer

Handles communication channels.

- Stdio: Direct process (Local, Fast)
- HTTP/SSE: Server-Sent Events (Remote, Auth support)



Data Layer

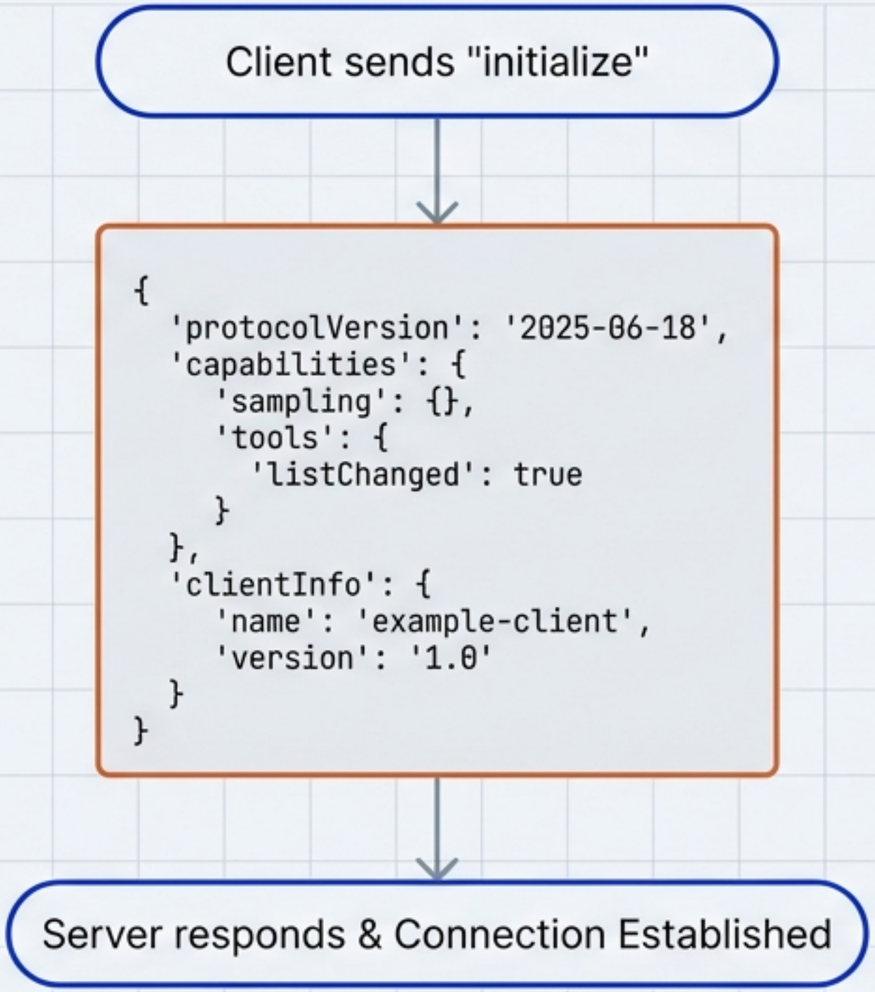
Handles logic and schema.

- JSON-RPC 2.0 Message Format
- Lifecycle & Errors
- Primitives (Tools, Resources)

The Transport layer abstracts the plumbing so the Data layer remains consistent anywhere.

Lifecycle Phase I: Initialization

The Handshake



Negotiation Points

1. Protocol Version (Must match)
2. Capabilities (What can we do?)
3. Identity (Who are you?)

Lifecycle Phase II: Tool Discovery

The Host asks “What can I do?” and the Server responds with a registry of capabilities.

`weather_current`

Description:
Returns current weather for a location. Use this when the user asks about temperature or conditions.

Input Schema (JSON):

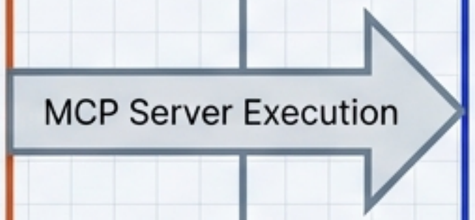
```
{
  type: 'object',
  properties: {
    location: { type: 'string' },
    unit: { type: 'string', enum: ['celsius', 'fahrenheit'] }
  },
  required: ['location']
}
```

This definition allows the LLM to ‘learn’ the tool instantly.

Lifecycle Phase III: Execution (The Loop)

The Request (tools/call)

```
{
  'method': 'tools/call',
  'params': {
    'name': 'weather_current',
    'arguments': {
      'location': 'San Francisco',
      'units': 'imperial'
    }
  }
}
```

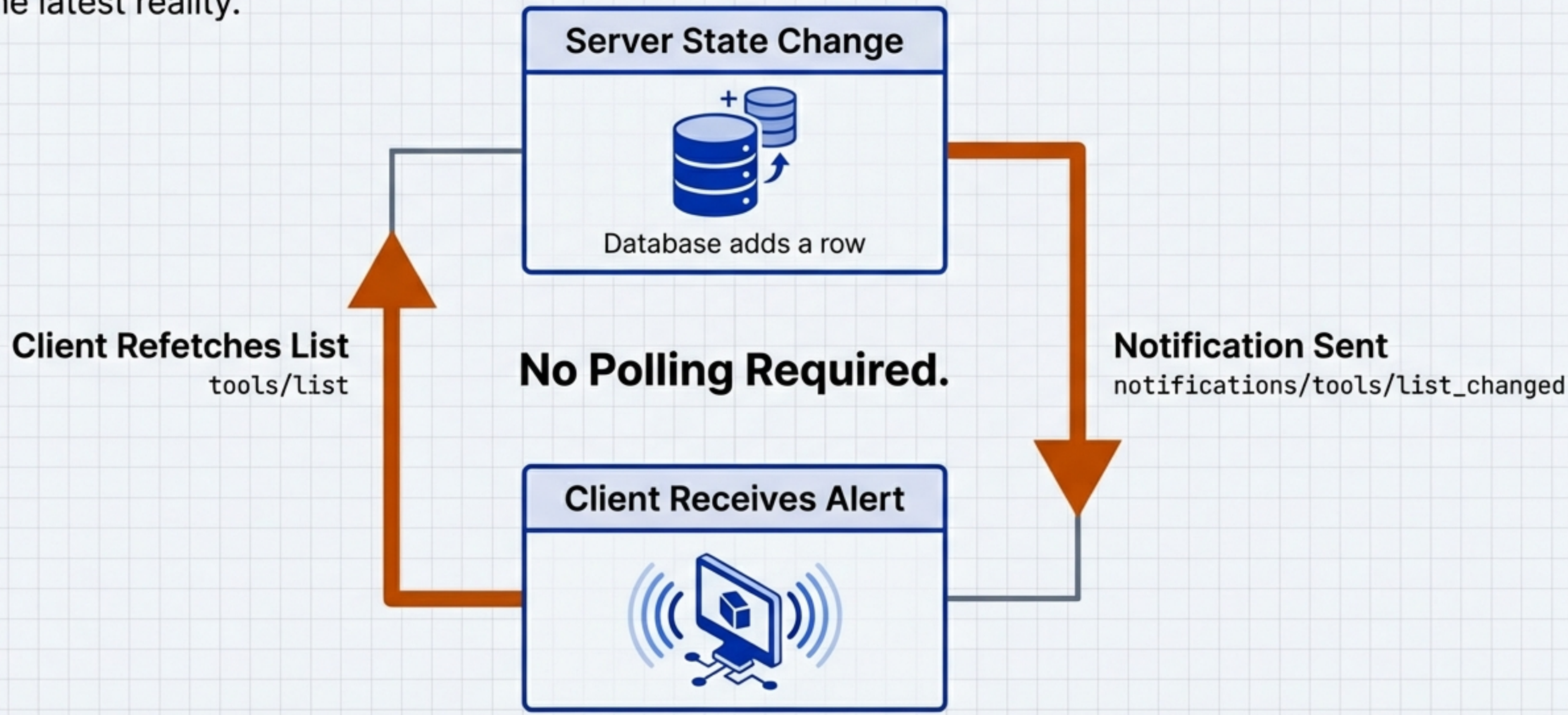


The Response

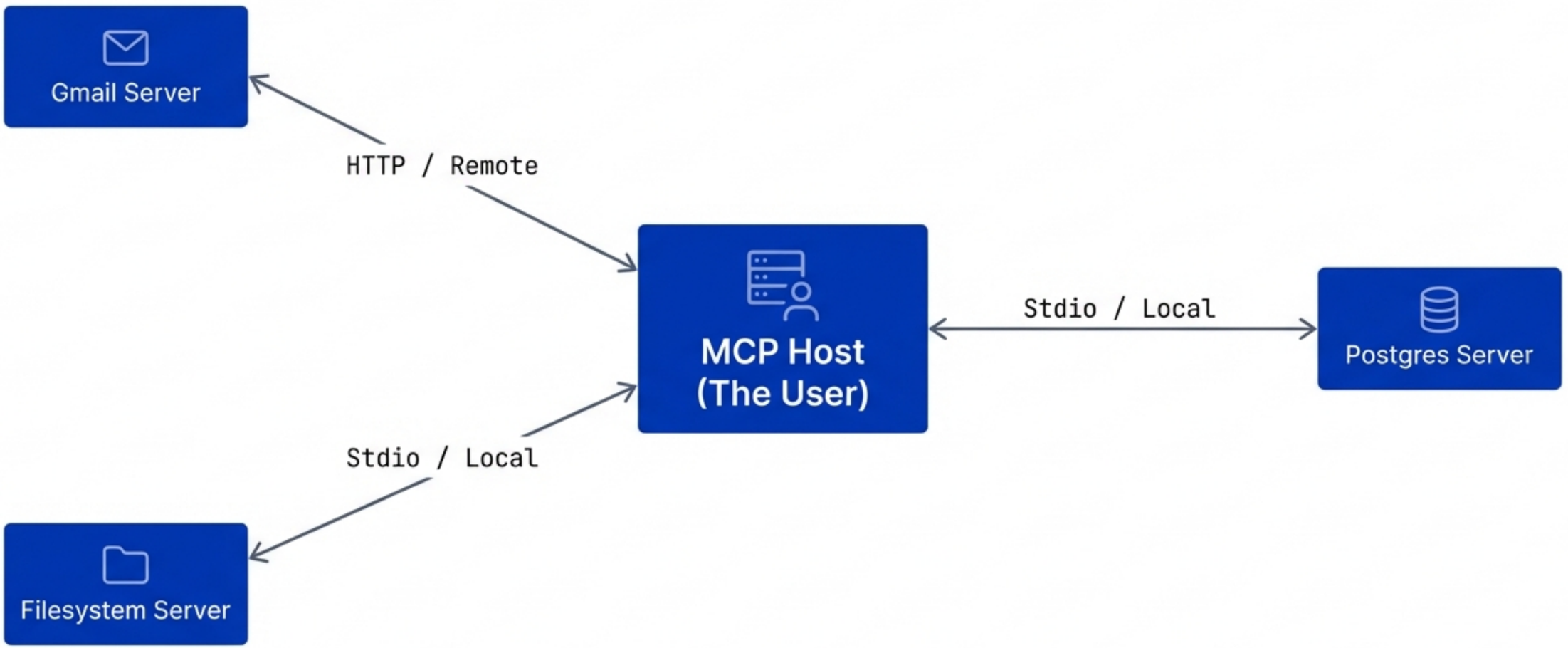
```
{
  'content': [
    {
      'type': 'text',
      'text': '72°F, Sunny, Wind NW 10mph'
    }
  ]
}
```

Keeping Context Synchronized

The world isn’t static. MCP servers push real-time updates to the Host, ensuring the AI always acts on the latest reality.

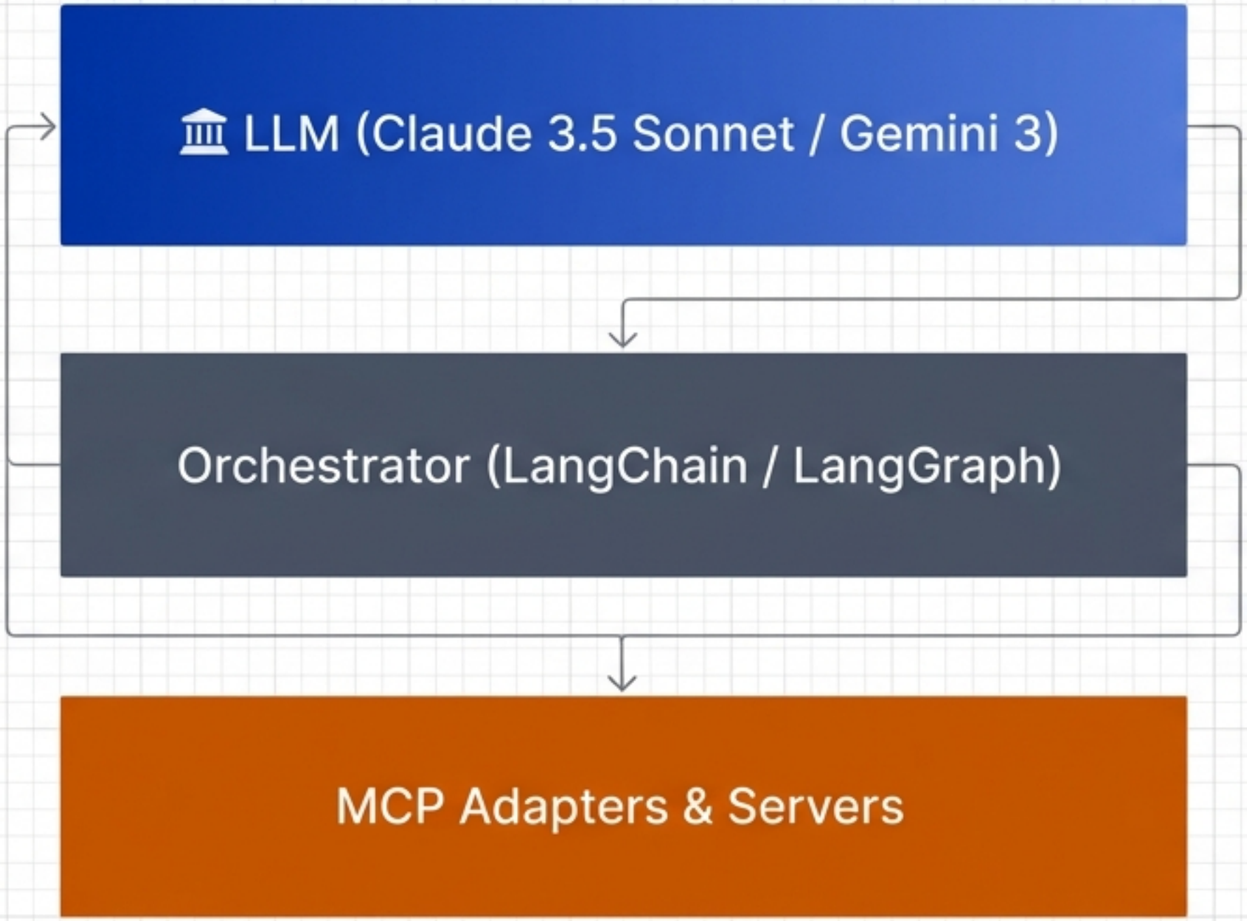


The Power of Multi-Server Architecture



The Host acts as an orchestrator, maintaining simultaneous connections to local and remote tools. The AI can read a local file, query a database, and email the results in a single session.

The Integration Stack




```
client = MultiServerMCPClient({
    'gmail': {
        'command': 'npx',
        'args': ['server-gmail']
    },
    'filesystem': {
        'command': 'uvx',
        'args': ['server-files']
    }
})
```

Connect pre-built servers with a few lines of configuration.

In Practice: The ‘Travel Agent" Workflow

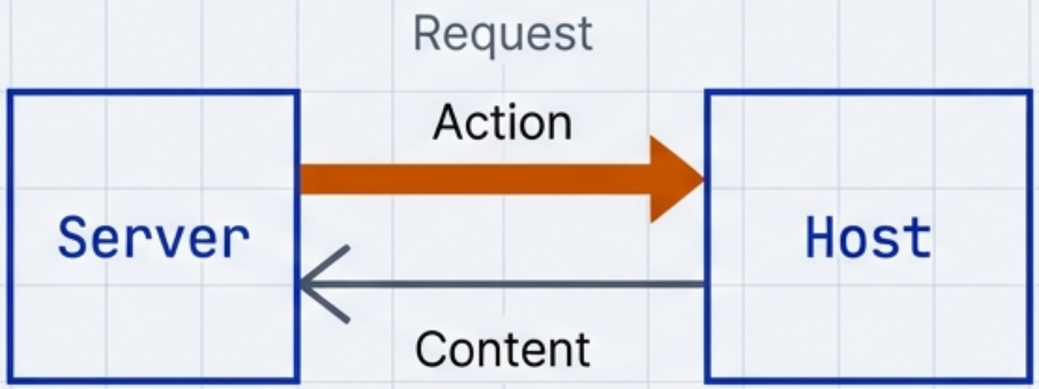
User Prompt: **“Plan a weekend trip to Paris.”**

Actionable outcomes across 3 platforms without leaving the chat.

Bidirectional Intelligence: Client Primitives

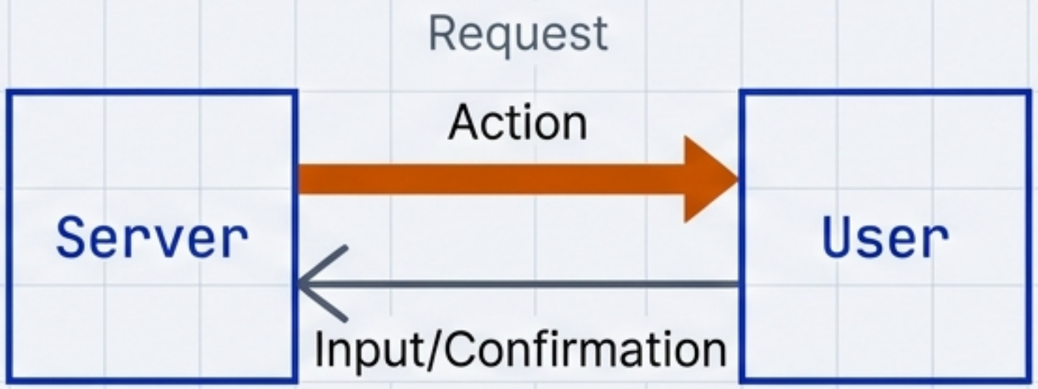
The conversation isn’t just one-way. Servers can leverage the Host’s capabilities.

Sampling



The Server asks the Host’s LLM to generate content (e.g., “Summarize this 50MB log file”).

Elicitation



The Server asks the User for input or confirmation (e.g., “Delete file? Y/N”).

Your Deployment Roadmap

Install Tools

- `uv` (Python manager)
- Node.js
- `fastmcp`

Pick a Host

- Use Claude Desktop for testing
- or build a LangChain App.

Connect a Server

- Run `npx @modelcontextprotocol/server-filesystem`

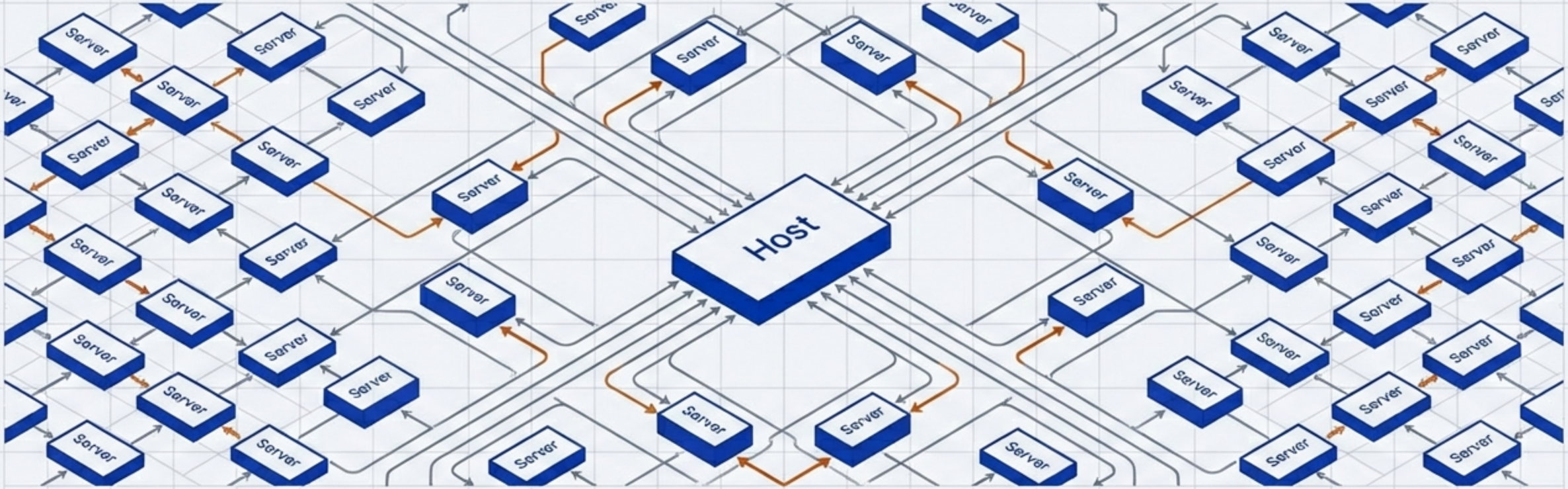
Configure

- Set environment variables
- and API keys.

Pro Tip:

Start with 'stdio' transport for local development. It's faster and requires requires no authentication.

The Future of Context



Standardized: Breaking the cycle of custom connectors.

Scalable: Plug-and-play architecture for thousands of tools.

Intelligent: Real-time, bi-directional, and secure.

Explore the specification at modelcontextprotocol.io