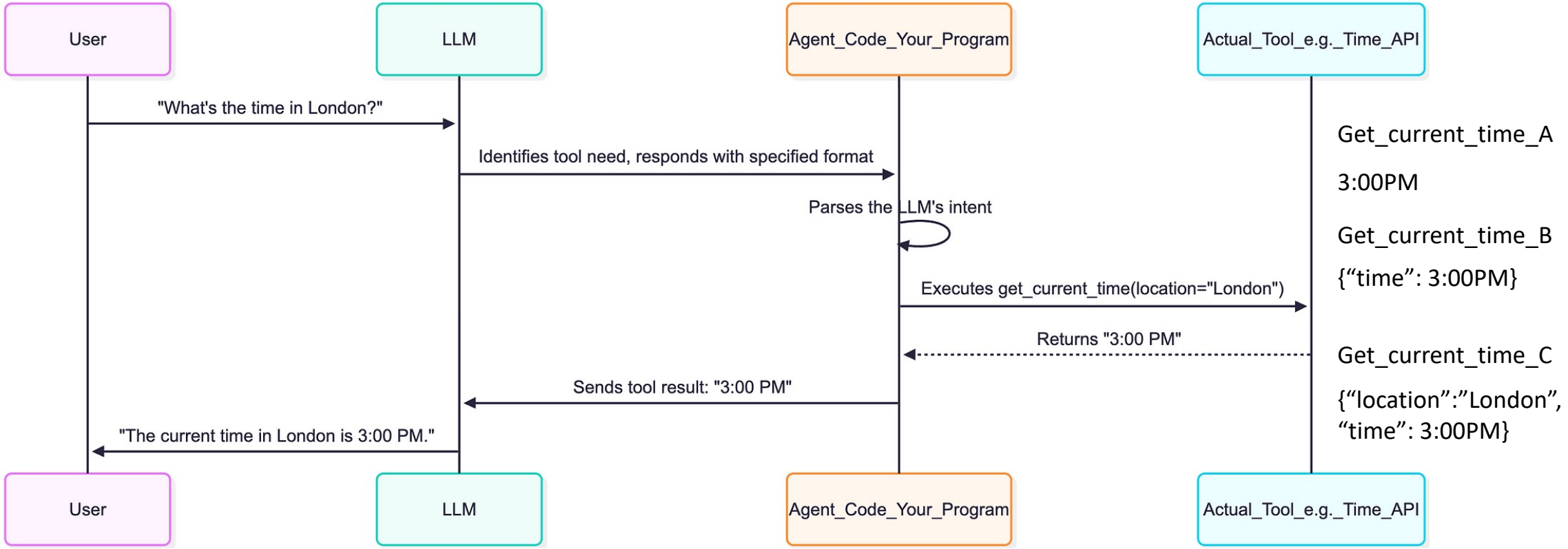
Model Context Protocol (MCP): Why, What and How

Yan Xu, Eashan Kaushik

Houston Machine Learning

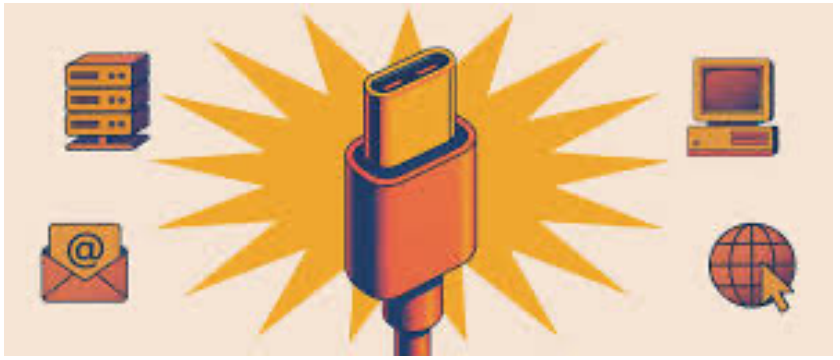
Why do we need MCP?

Growing pain: Everyone doing their own tools for LLM



LLM tool interaction without MCP

Why do we need MCP?



The Model Context Protocol (MCP) aims to bring some order to the chaos. We need a universal USB-C standard for LLM tools!

Standardization

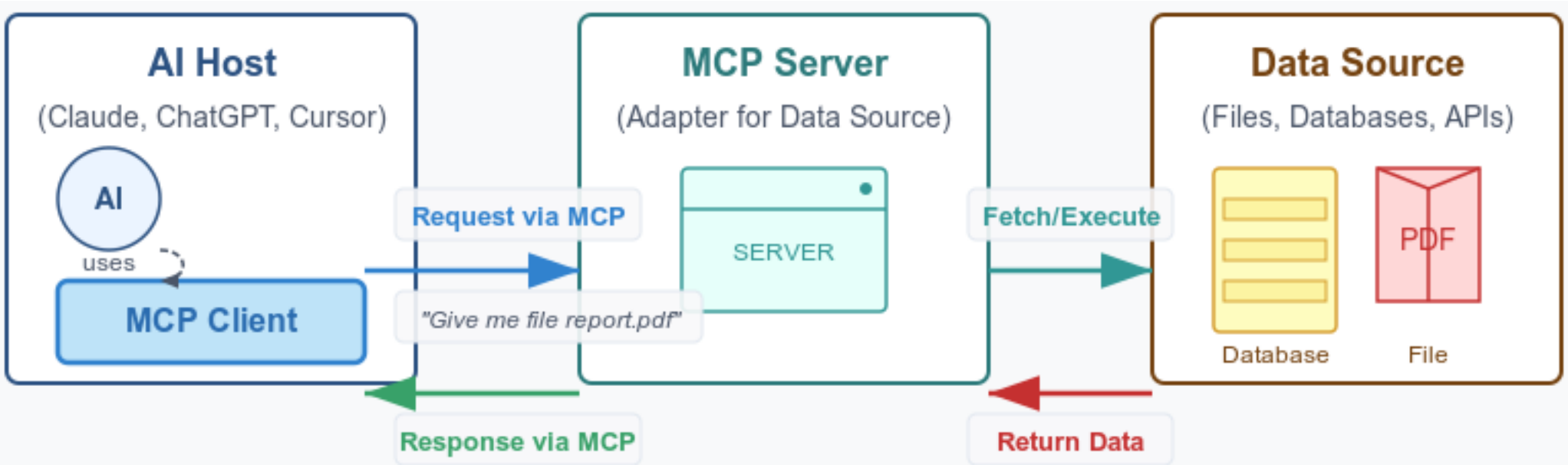
Flexibility and Scalability

Enhance capabilities

Timeline: Power of Scale

- **November 2024: Anthropic** introduces and open-sources the Model Context Protocol (MCP) as a standard for connecting AI assistants to data systems.
- **January 2025: Zed** editor, **Cline**, and **Cursor** add MCP support.
- **February 2025: Anthropic** launches **Claude Code** with MCP support. Over 1,000 open-source connectors emerge.
- **March 2025:** Cloudflare and Sentry release guides for deploying a production-ready remote MCP server. **OpenAI** officially adopts the MCP.
- **April 2025: VS Code** adds MCP support. **GitHub** launches its official MCP server in public preview.
- **May 2025: Microsoft** announces general availability of MCP integration in Copilot Studio.
- **May 2025: AWS** introduces Serverless MCP Server
- **June 2025:** Remote MCP support is added to **Claude Code**.

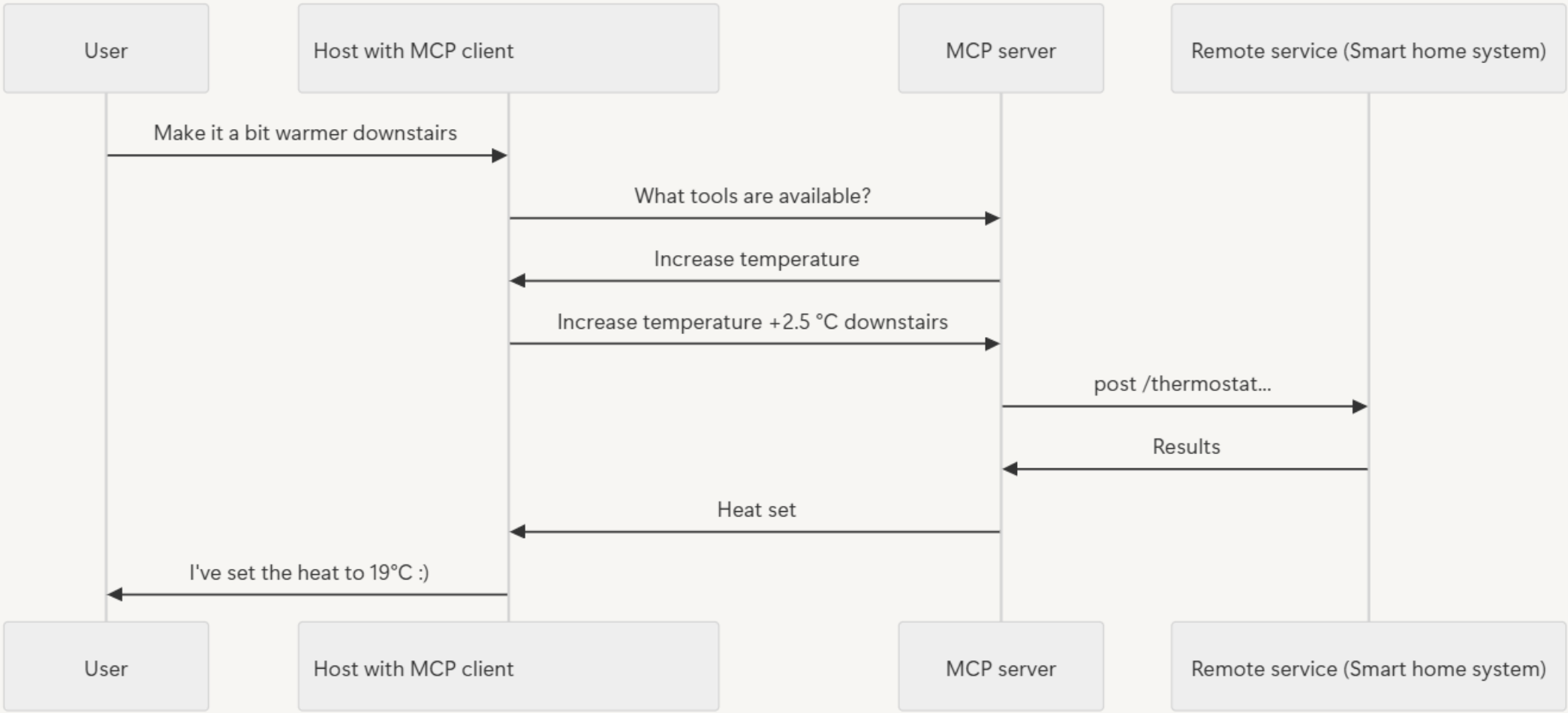
MCP Architecture



Model Context Protocol (MCP) Flow

The MCP Client translates AI requests into the standardized protocol format, communicates with MCP Servers, which then interact with external Data Sources.

MCP use in a real-life example



MCP communication: Format

At its core, MCP uses **JSON-RPC 2.0** (Remote Procedure Call) as the message format for all communication between Clients and Servers. JSON-RPC is a lightweight remote procedure call protocol encoded in JSON. Streamable HTTP/Server-Sent Events (SSE) is used for communication across networks.

Request

```
{
  "jsonrpc": "2.0",
  "id": 1,
  "method": "tools/call",
  "params": {
    "name": "weather",
    "arguments": {
      "location": "San Francisco"
    }
  }
}
```

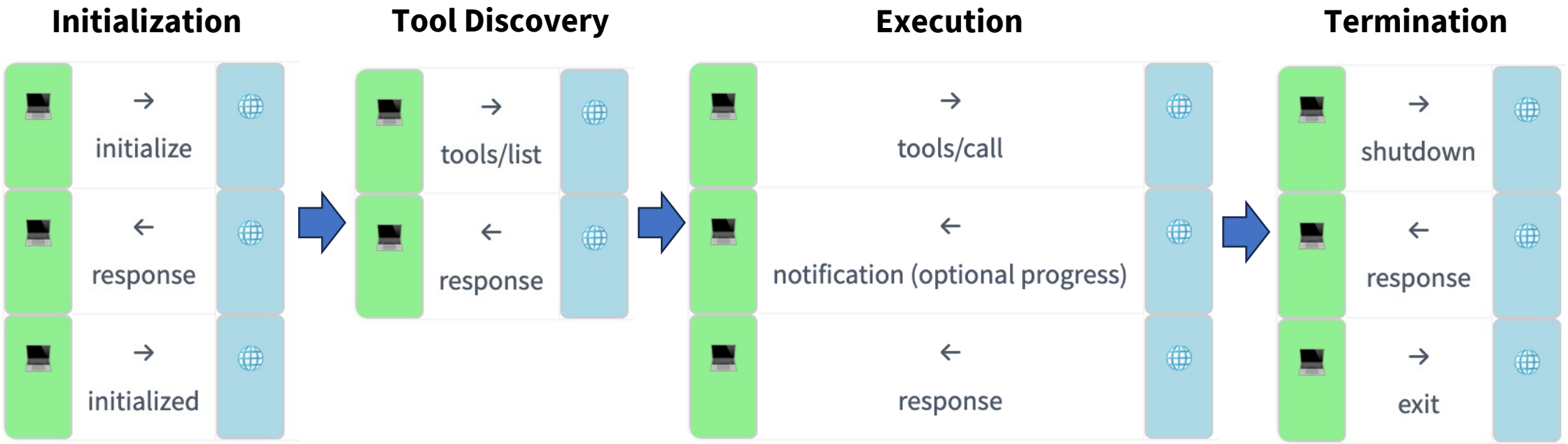
Response

```
{
  "jsonrpc": "2.0",
  "id": 1,
  "result": {
    "temperature": 62,
    "conditions": "Partly cloudy"
  }
}
```

Error handling

```
{
  "jsonrpc": "2.0",
  "id": 1,
  "error": {
    "code": -32602,
    "message": "Invalid location parameter"
  }
}
```

MCP communication: Interaction Lifecycle



AI Host: LLM prompt format

****System prompt:****

You are a helpful AI assistant that specializes in leveraging external tools to answer user requests. Your primary goal is to accurately interpret user requests, identify the most appropriate tools from the provided list, and provide the necessary parameters to invoke those tools. You should then present the results of the tool execution to the user in a clear and concise manner.

****User Request:****

[User's natural language request]

****Available Tools:****

[List of tools with descriptions, parameters, and return types]

****In context examples:****

[List of examples of how to use the tools]

****Instructions:****

Analyze the user request

Select the most relevant tool from the "Available Tools" list to fulfill the user’s request.

Identify the necessary input parameters for the chosen tool based on the user's request.

If a tool is selected, output the tool name and its parameters in a structured format (e.g., JSON).

MCP server

Resources

Integrate all files or content the server can provide to AI

Tools

Run a command line, modify data, search web, call 3rd party API etc.

Prompts

Reusable prompt templates or workflows

Sampling

Servers to request LLM completions through the client

Resources

Resources represent any kind of data that an MCP server wants to make available to clients. This can include:

- Text resources
  - Source code
  - Configuration files
  - Log files
  - JSON/XML
  - Plain text
- Binary resources encoded in base64
  - Images
  - Pdfs
  - Audio/Video

Resources

- URI: [protocol]://[host]/[path]
  - file:///home/user/documents/report.pdf
  - postgres://database/customers/schema
  - screen://localhost/display1
- Each resource includes:

```
{
  uri: string;              // Unique identifier for the resource
  name: string;             // Human-readable name
  description?: string;     // Optional description
  mimeType?: string;        // Optional MIME type
  size?: number;            // Optional size in bytes
}
```

Reusable Prompts

```
// Request
{
  method: "prompts/list";
}

// Response
{
  prompts: [
    {
      name: "analyze-code",
      description: "Analyze code for potential improvements",
      arguments: [
        {
          name: "language",
          description: "Programming language",
          required: true,
        },
      ],
    },
  ];
}
```

```
// Request
{
  method: "prompts/get",
  params: {
    name: "analyze-code",
    arguments: {
      language: "python"
    }
  }
}

// Response
{
  description: "Analyze Python code for potential improvements",
  messages: [
    {
      role: "user",
      content: {
        type: "text",
        text: "Please analyze the following Python code for potential improvements:\
      }
    }
  ]
}
```

Tools: Boundless possibilities with 3rd party vendors

ATLASSIAN

Connect to Jira, Confluence, and other Atlassian tools to manage issues, access

CLOUDFLARE

Manage Workers deployments, D1 databases, R2 storage, and KV stores directly through

INTERCOM

Manage customer conversations, access support tickets, retrieve customer

Linear

Manage Linear issues, projects, and team workflows through Claude. Create and update

PayPal

Interact with PayPal's payment ecosystem to process transactions, manage

PLAID

Access bank account data, transaction history, and financial information through Plaid's secure API. Verify

zapier

Search knowledge across your apps and execute real-world actions by connecting Claude with the 8,000 apps on Zapier.

invideo

Transform text prompts into full-length videos with AI-generated scripts, visuals, voiceovers, and subtitles.

Square

Access Square's commerce platform to view transaction data, manage customer profiles, track inventory, process payments, and analyze sales

Tools

```
{
  name: string;           // Unique identifier for the tool
  description?: string;   // Human-readable description
  inputSchema: {          // JSON Schema for the tool's parameters
    type: "object",
    properties: { ... }  // Tool-specific parameters
  },
  annotations?: {         // Optional hints about tool behavior
    title?: string;       // Human-readable title for the tool
    readOnlyHint?: boolean;    // If true, the tool does not modify its environment
    destructiveHint?: boolean; // If true, the tool may perform destructive updates
    idempotentHint?: boolean;  // If true, repeated calls with same args have no adc
    openWorldHint?: boolean;   // If true, tool interacts with external entities
  }
}
```

Tools: Local system operation

```
{
  name: "execute_command",
  description: "Run a shell command",
  inputSchema: {
    type: "object",
    properties: {
      command: { type: "string" },
      args: { type: "array", items: { type: "string" } }
    }
  }
}
```

Tools: API integration

```
{
  name: "github_create_issue",
  description: "Create a GitHub issue",
  inputSchema: {
    type: "object",
    properties: {
      title: { type: "string" },
      body: { type: "string" },
      labels: { type: "array", items: { type: "string" } }
    }
  }
}
```

Sampling: Server to request from Client

- Sampling is a powerful MCP feature that allows servers to request LLM completions through the client, enabling sophisticated agentic behaviors. This human-in-the-loop design ensures users maintain control over what the LLM sees and generates.

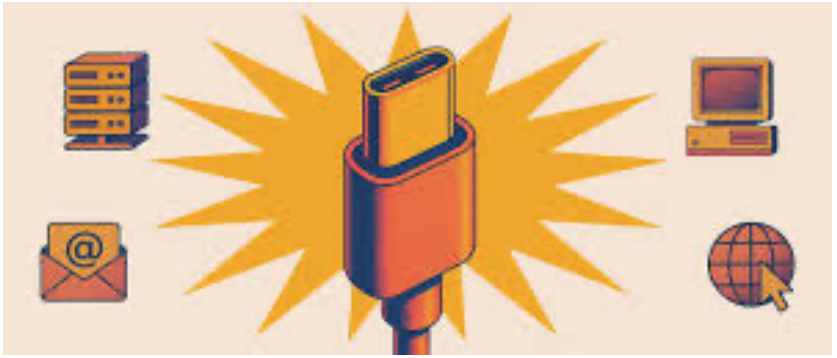
```
{
  "method": "sampling/createMessage",
  "params": {
    "messages": [
      {
        "role": "user",
        "content": {
          "type": "text",
          "text": "What files are in the current directory?"
        }
      }
    ],
    "systemPrompt": "You are a helpful file system assistant.",
    "includeContext": "thisServer",
    "maxTokens": 100
  }
}
```

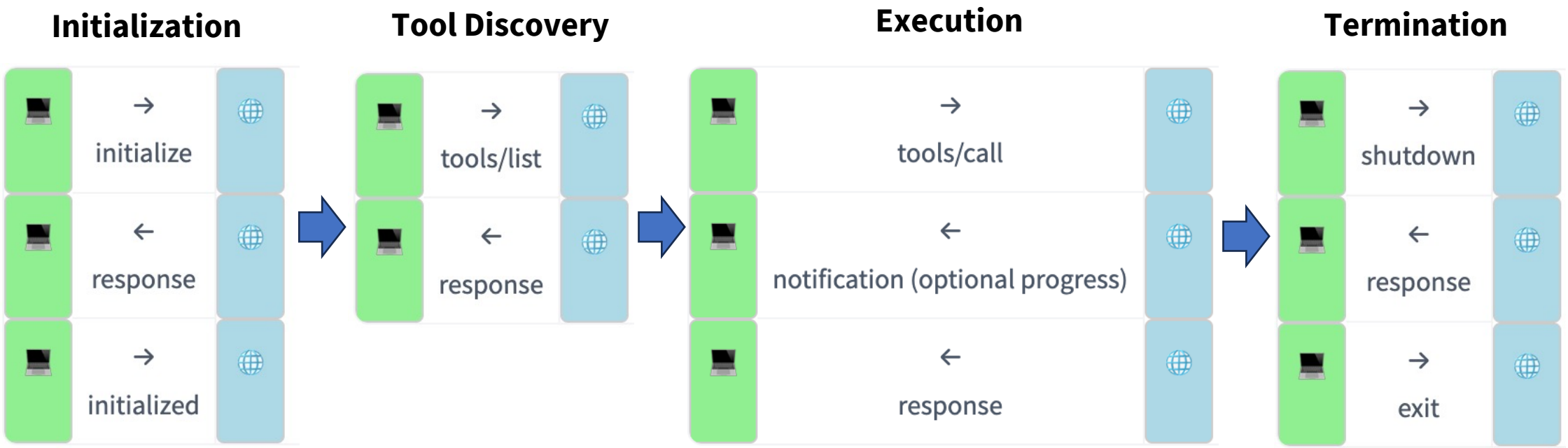
The client returns a completion result:

```
{
  model: string,  // Name of the model used
  stopReason?: "endTurn" | "stopSequence" | "maxTokens" | string,
  role: "user" | "assistant",
  content: {
    type: "text" | "image",
    text?: string,
    data?: string,
    mimeType?: string
  }
}
```

Recap

MCP is the USB standard for LLM tools!






DEMO TIME
makeameme.org

Upcoming meetups – Open to proposals and guest speakers!

- AI agent real-world use cases
- Build AI agents with MCP
- Hands-on sessions
- Panel discussions

Slides posted at:
https://github.com/YanXuHappygela/LLM-reading-group

Recordings posted at:

YanAITalk

@yanaitalk · 3.04K subscribers · 68 videos

Make machine learning easy to understand! ...more