# Welcome

Introductions
What Is This Course About?

# Who Am I?

**Paulo Dichone**

Software, Cloud, AI Engineer
and Instructor

# What Is This Course About?

- MCP - Model Context Protocol
  - What is it?
  - How it works?
  - Hands-on - Build MCP servers
  - Test MCP Servers locally
  - Deploy and test MCP servers remotely

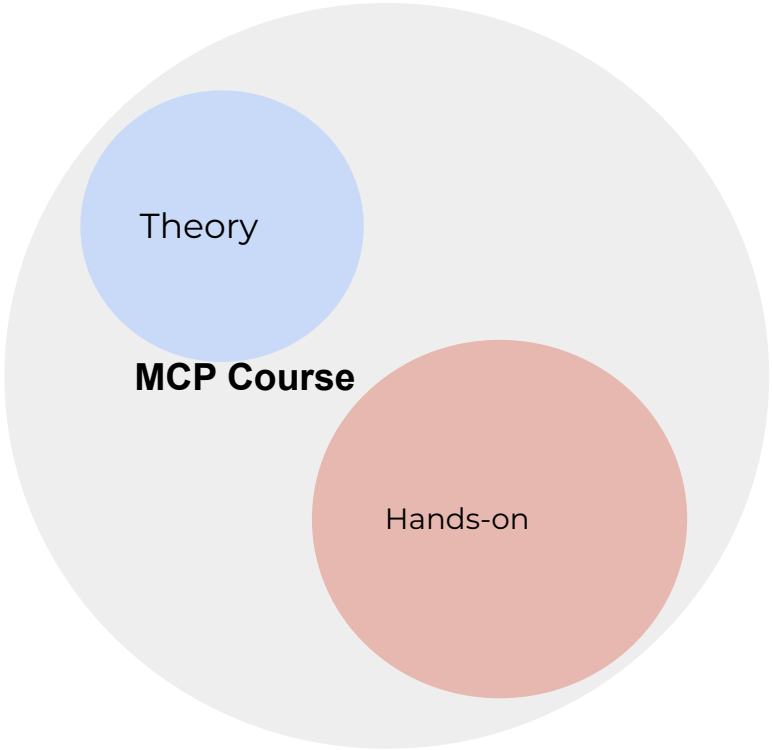
# Course Prerequisites

1. Know Python (basics at least); will not teach how to to code
2. Fundamentals of LLMs and AI
3. Be willing to learn new skills

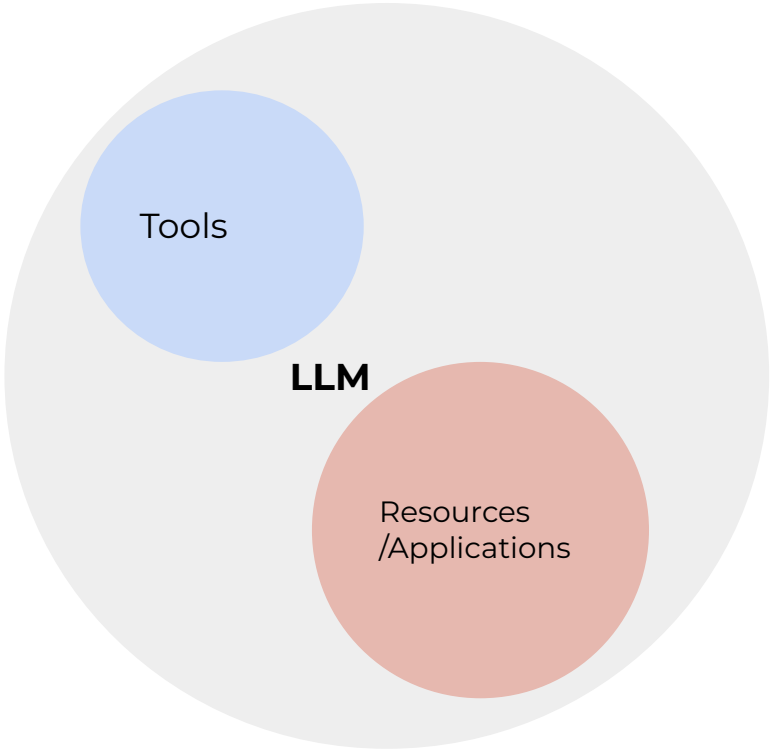
# What You'll build...

## Demo...

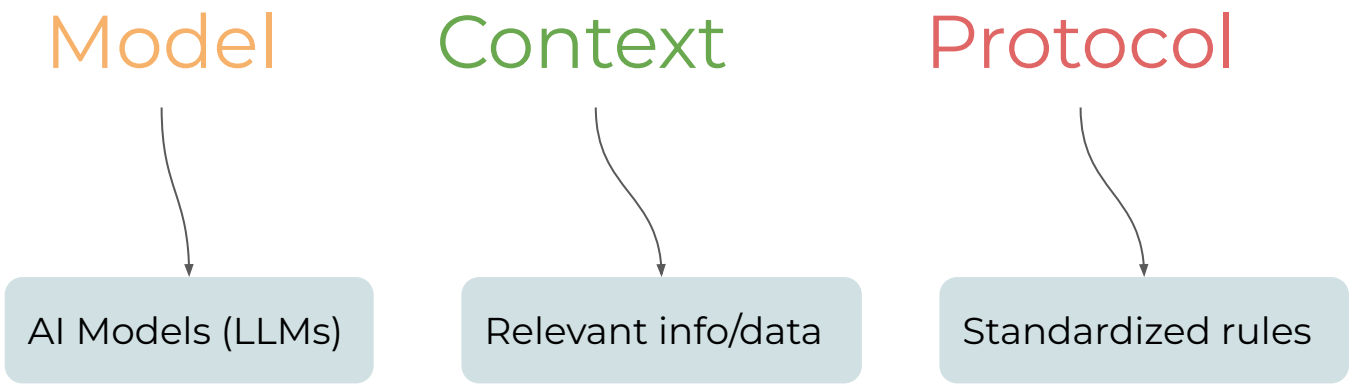
# Course Structure



# MCP - Model Context Protocol

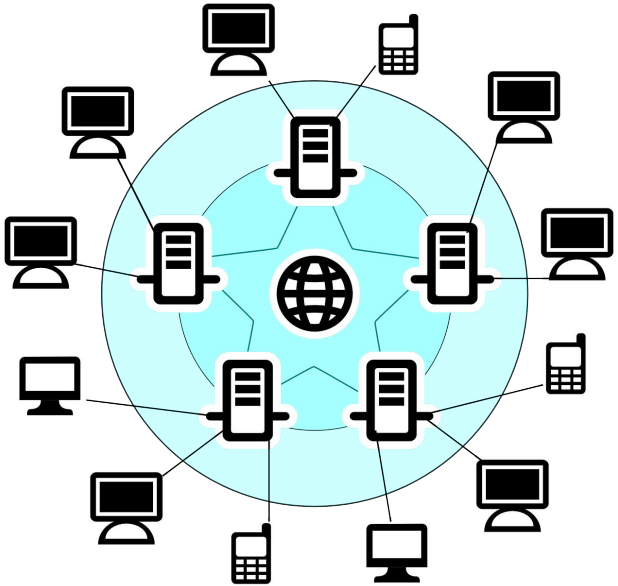


# Motivation



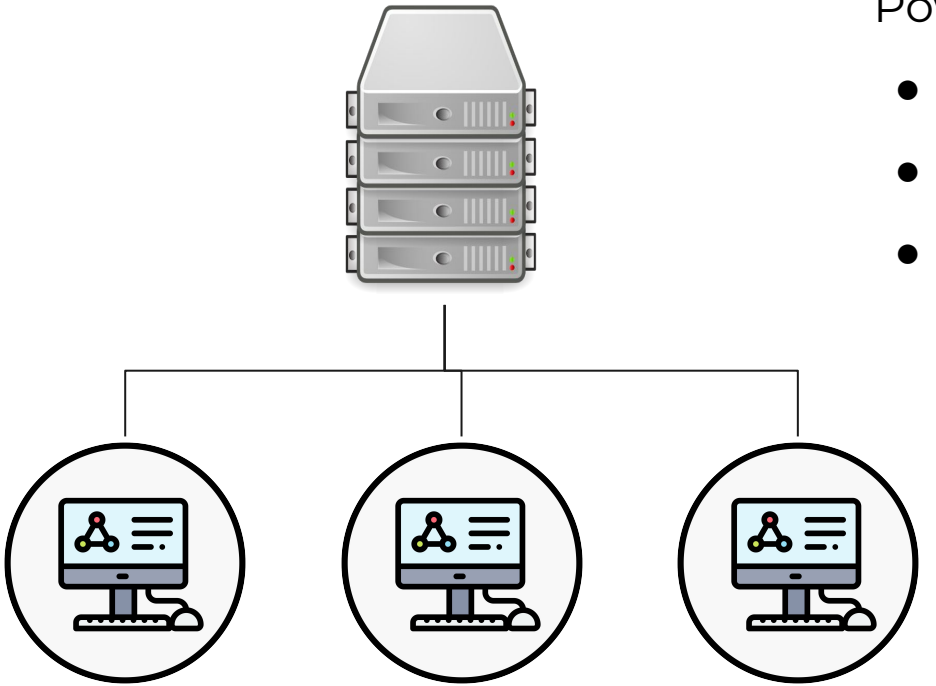

... is a ***standardized*** set of rules that allows AI models to access and utilize external information and tools; expanding training.

# The Digital Dialogue: How Computers Communicate



# Servers and Clients

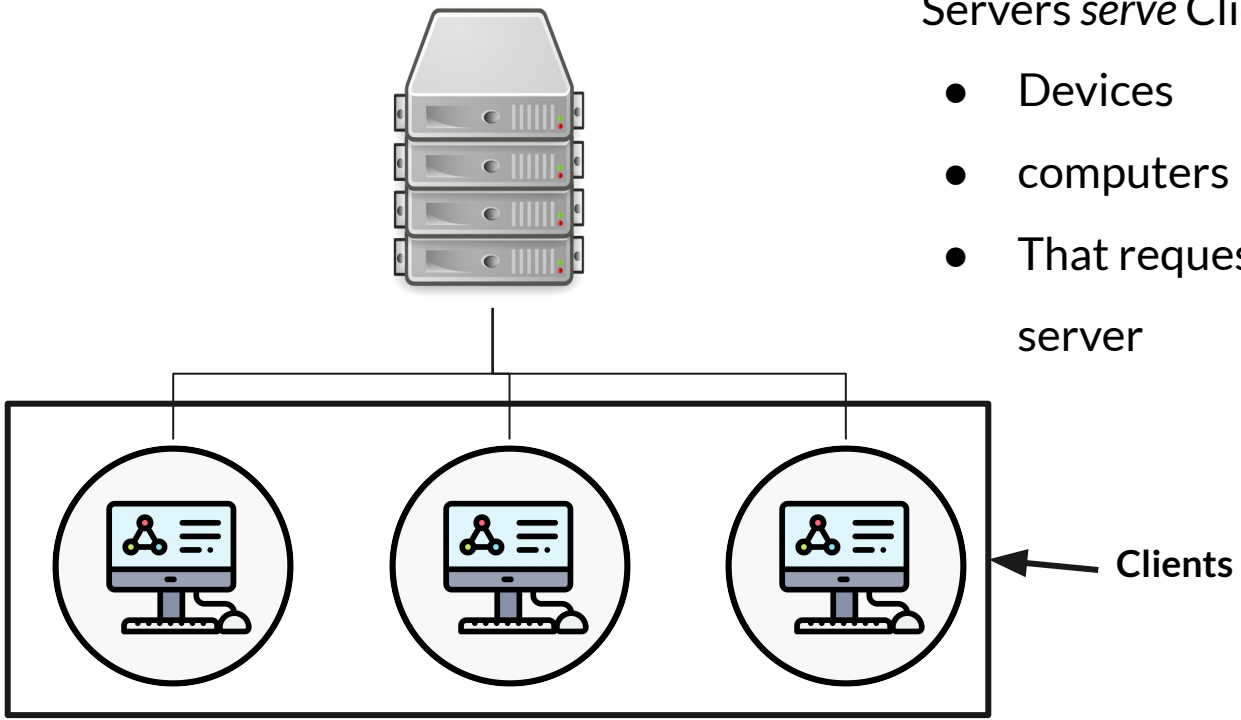
# Servers



Powerful computers that provide:

- Resources
- Services
- Data to other computers or devices

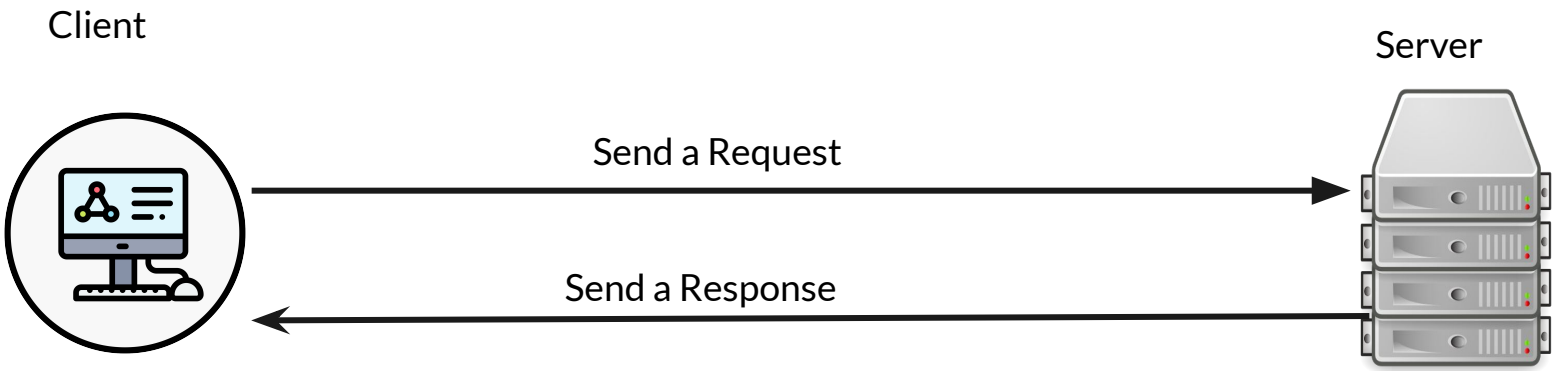
# Servers & Clients




Servers *serve* Clients which are

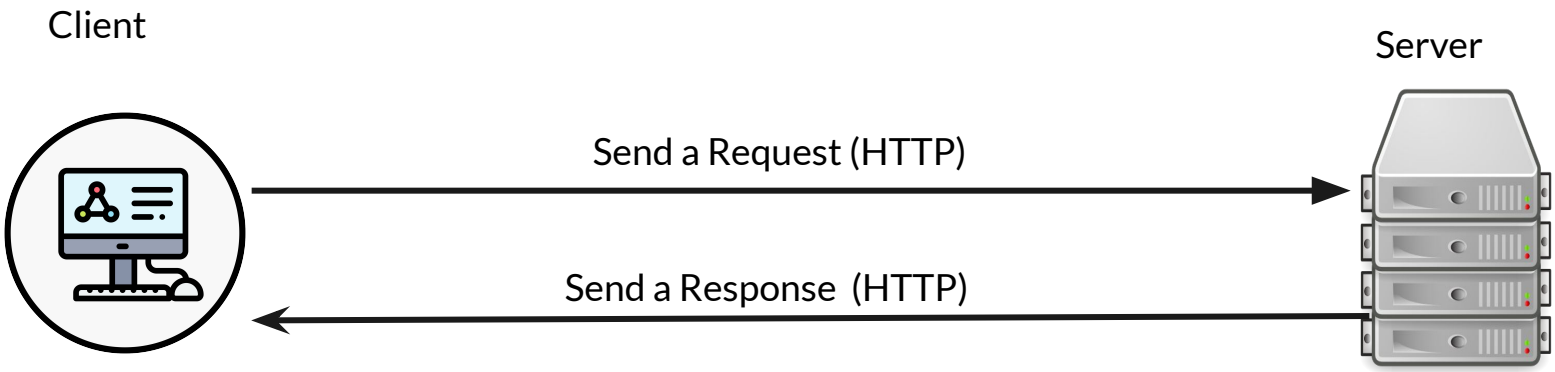
- Devices
- computers
- That request resources, data from the server

# Client-Server Communication




*Listens for incoming request(s)*

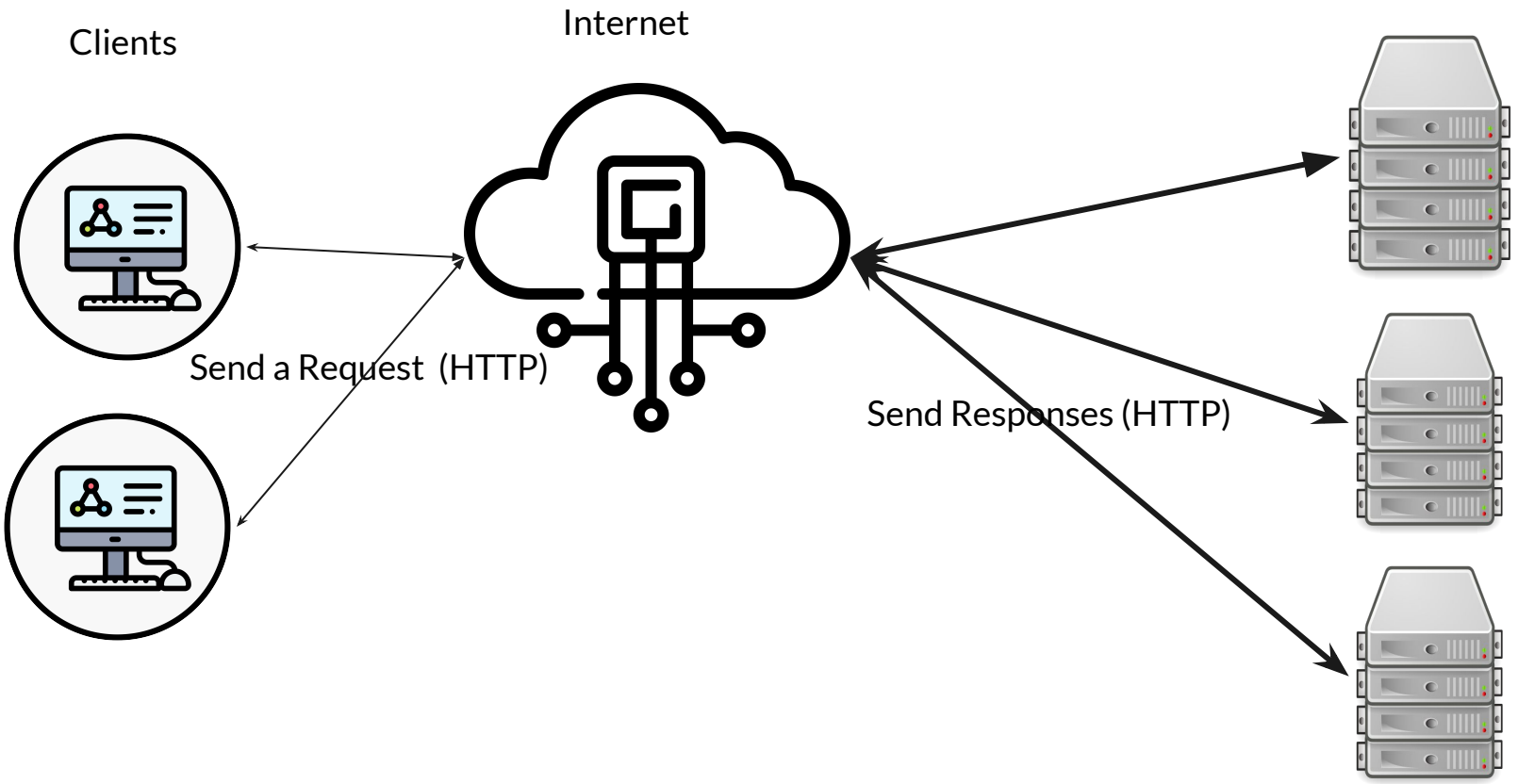
# Client-Server Communication Protocol



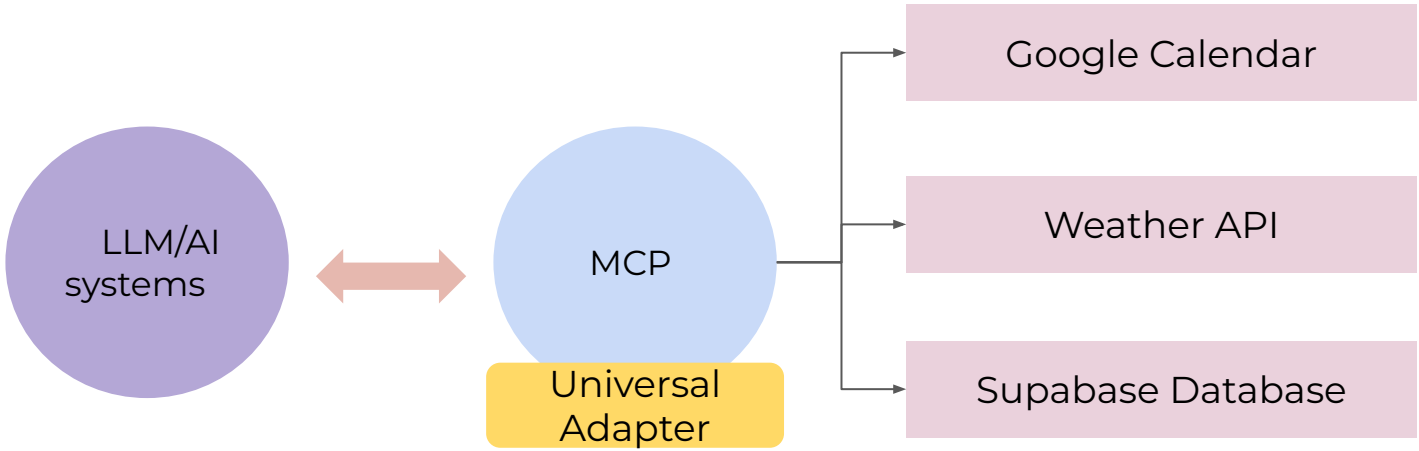

Protocols:

- HTTP - Hypertext Transfer Protocol (transfer text, images, audio and other multimedia
- FTP - File Transfer Protocol
- ...

# Client-Server Communication Protocol - Internet



# MCP: The Universal Adapter for AI



# MCP: The Universal Adapter for AI

## 🔌 Before MCP

**Custom Integrations Chaos**

Custom API | Proprietary | Brittle Code | One-off

`M AI Models × N Services = M×N Custom Connectors`

### 🧯 The Integration Nightmare

Every AI model needed custom code for each service. Fragile, error-prone, and impossible to maintain at scale.

## ✨ With MCP

**Universal Standard**

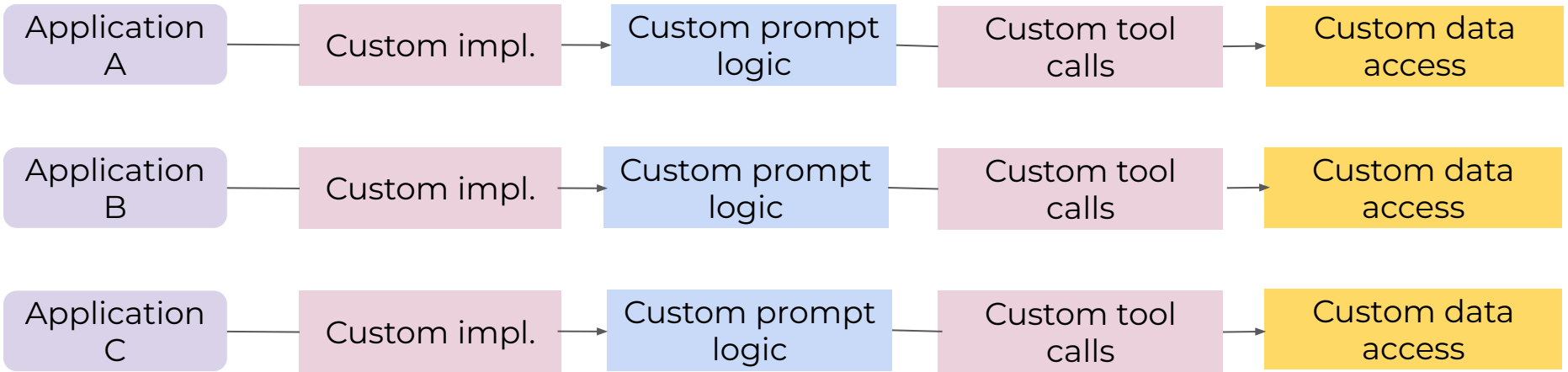
🔌 MCP Protocol

`M AI Models + N Services = M+N MCP Connections`

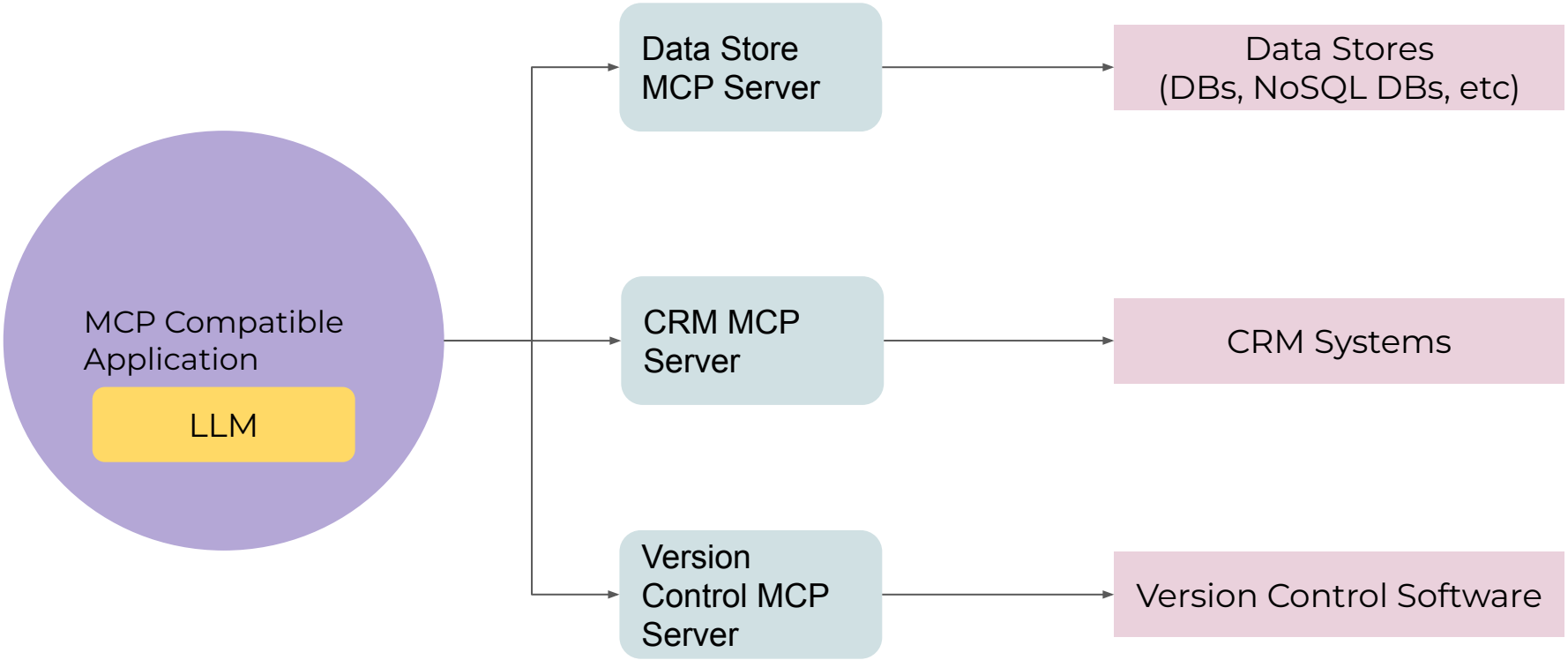
### 🎯 Universal Integration

One standard protocol connects any AI to any service. Simple, reliable, and infinitely scalable.

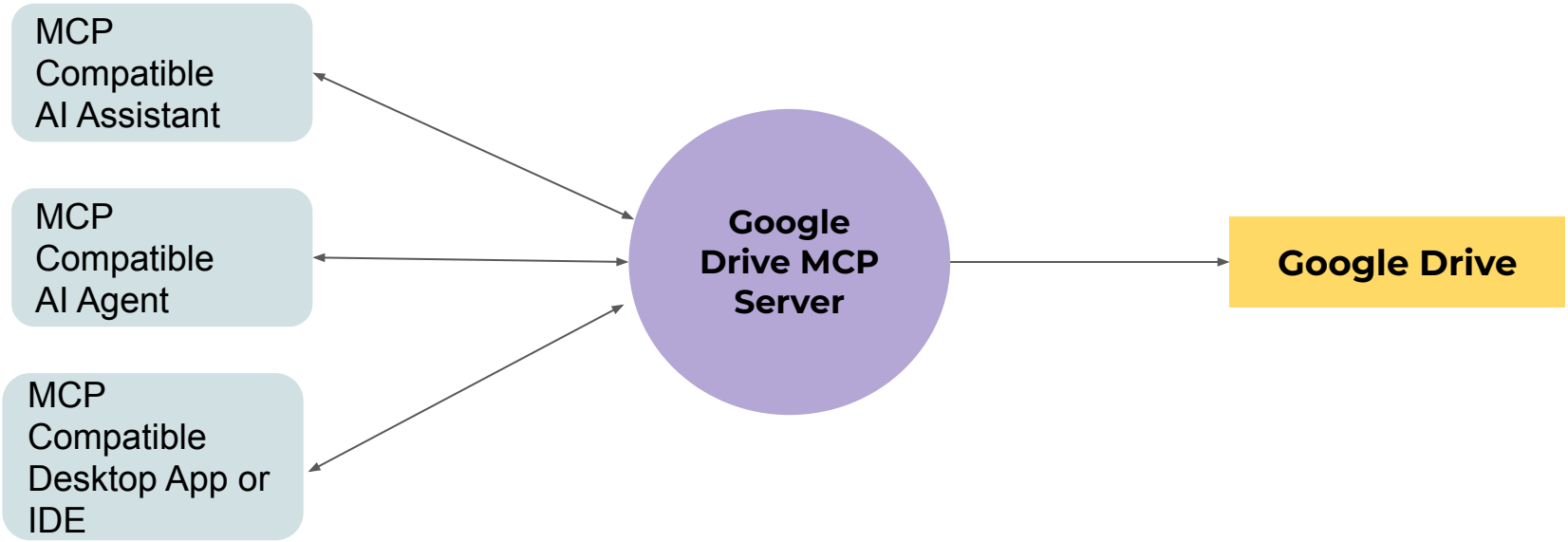
# Without MCP: Fragmented AI Development



# With MCP: Standardized AI Development



# With MCP: MCP servers are reusable...



# The Problems MCP Solves for LLMs

- **Knowledge cutoffs**: LLMs only know what they know
- **Hallucinations**: MCP enables LLMs to access more “context” from other sources.
- **Isolated Intelligence**: LLMs can’t natively interact with external systems, perform actions or access private user data.
- **Complex and Brittle Integrations:** Before MCP, developers would need to build custom, fragile integrations for each service.

# Advantages of MCP for LLMs and AI Agents

- **Enabling AI Agents:** MCP provides a standardized way for these agents to discover and utilize information and perform actions beyond what they can do.
- **Personalization:** securely access user-specific (with appropriate permissions)
- **Specialized Knowledge:** LLMs can tap into domain-specific knowledge bases and tools which provides them with expert-level responses.
- **Enhanced Security:** robust security controls.

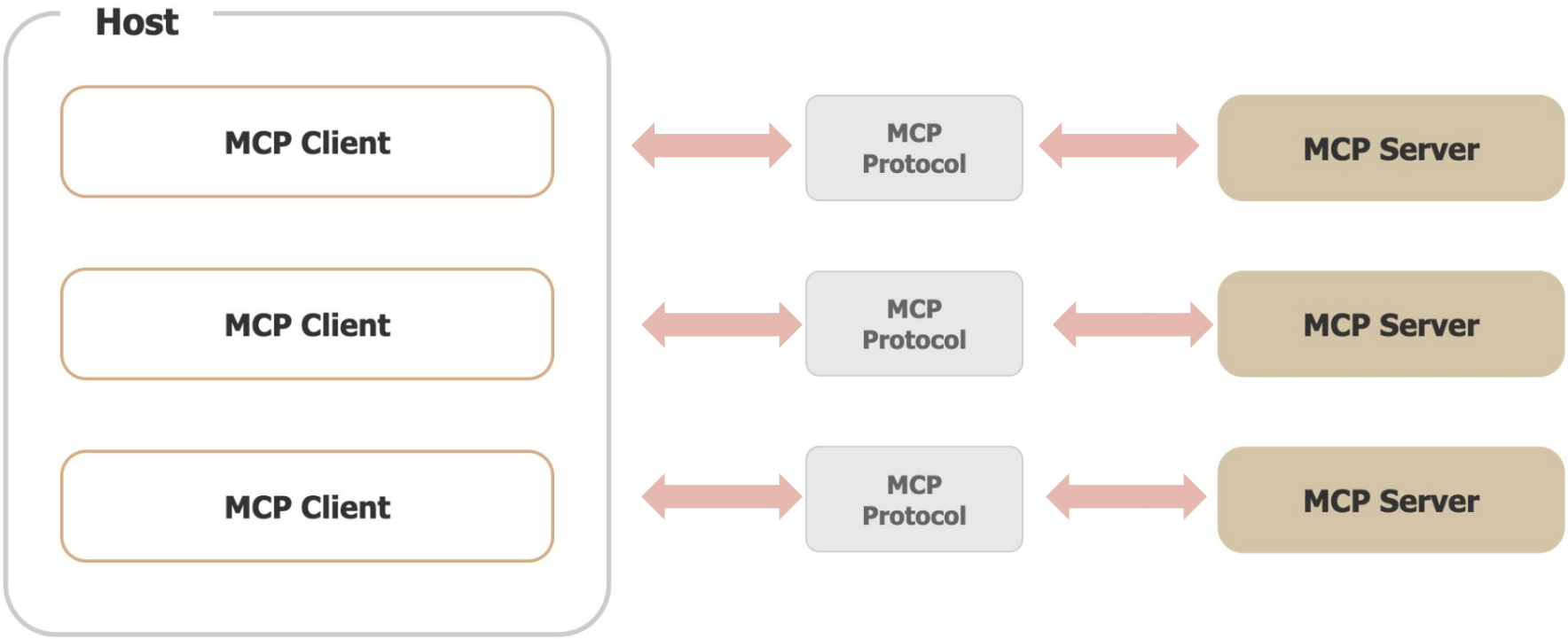
# With MCP: Standardized AI Development

- For AI application developers
  - Can connect AI apps to any MCP server with minimal (or zero) additional work.
- For tool or API Developers
  - Build once, and it can be adopted everywhere
- For AI applications users
  - AI applications have extensive capabilities
- For enterprises
  - Clear separation of concerns between AI products teams

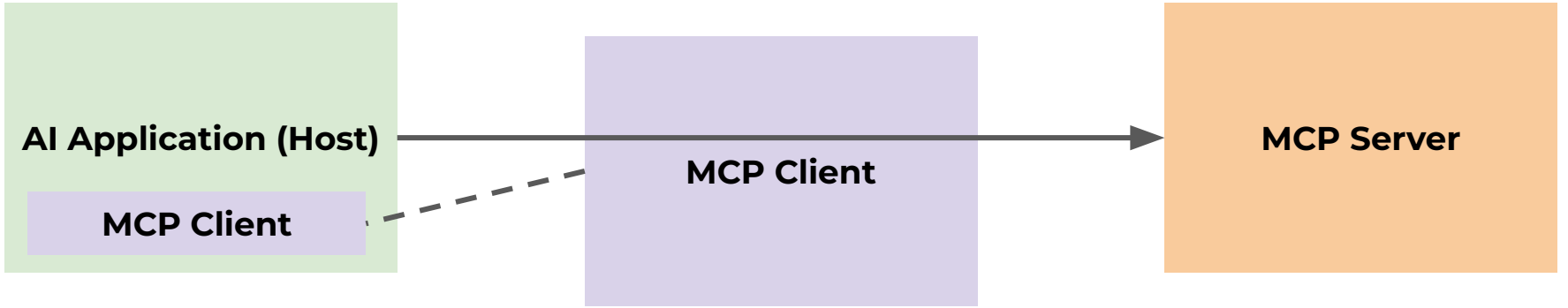
# Common Questions

| | |
|---|---|
| **Who authors the MCP Server?** | Anyone! Often the service provider itself will make their own MCP implementation. You can make a MCP server to wrap up access to some service. |
| **How is using an MCP Server different from just calling a service's API directly?** | MCP Servers provide tool schemas + functions.<br>If you want to directly call an API directly, you'll be authoring those on your own. |
| **Sounds like MCP Servers and tool use are the same thing.** | MCP Servers provide tool schemas + functions already defined for you. |

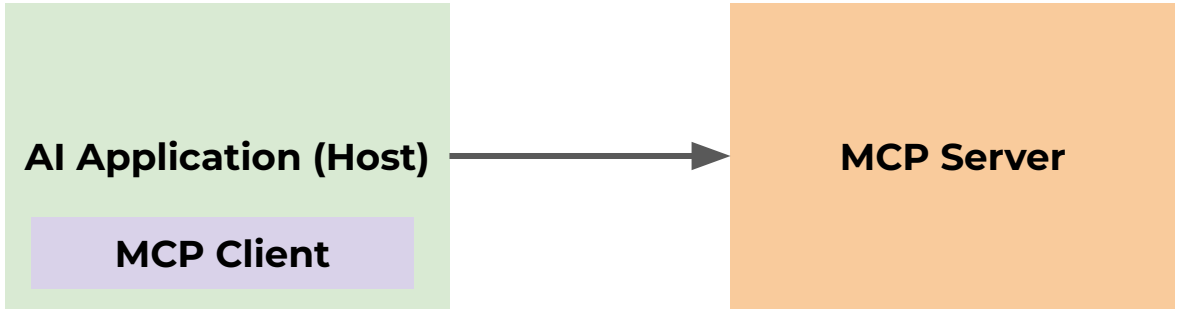
# MCP Architecture



# MCP - MCP Client Component



# MCP - MCP Server Deep Dive



# MCP Server Core Architecture

## Protocol Handler

Manages JSON-RPC 2.0 communication, message routing, and capability negotiation

## Transport Layer

Handles different communication methods: STDIO, HTTP+SSE, or custom transports

## Capability Engine

Implements Resources, Tools, and Prompts based on server's declared capabilities

## Security Layer

Authentication, authorization, request validation, and data protection

# Client-Server Architecture

### Host

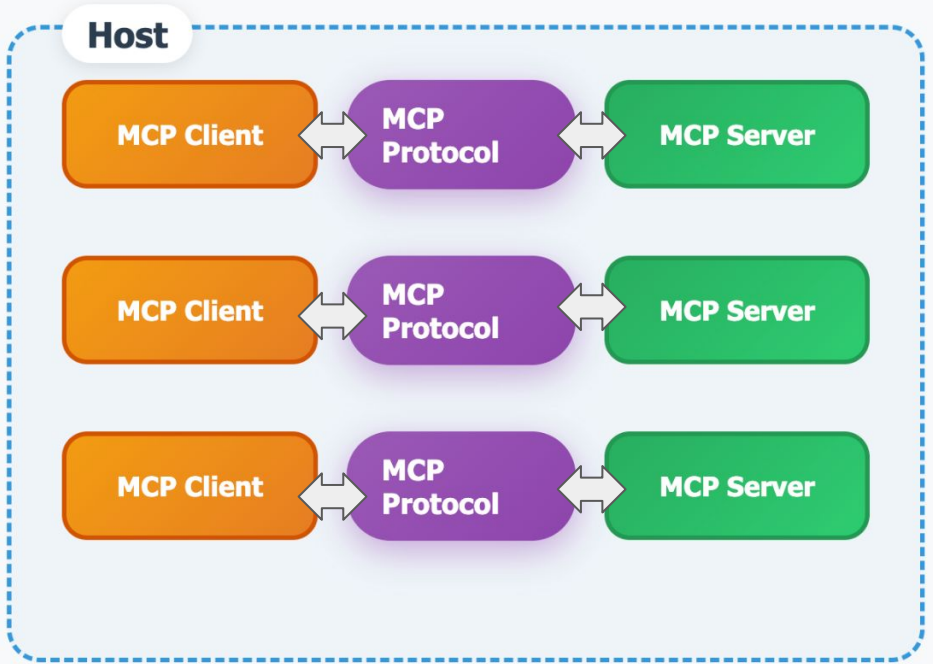
are LLM applications that want to access data through MCP (ex: Claude Desktop, IDEs, AI agents).

### MCP Servers

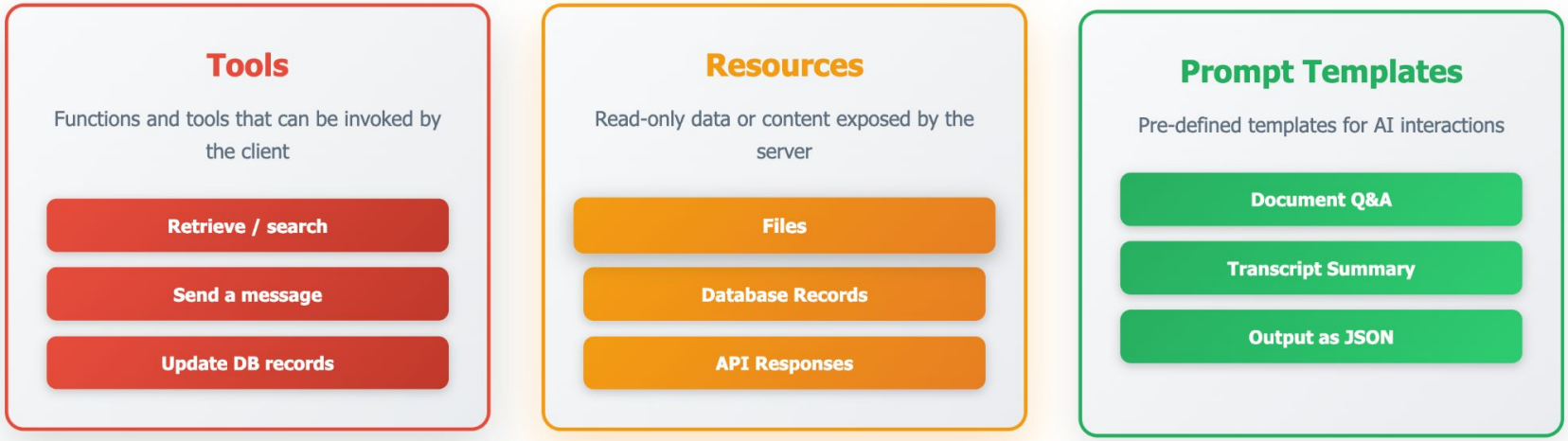
are lightweight programs that each expose specific capabilities through MCP.

### MCP Clients

maintain 1:1 connections with servers, inside the host application.



# How Does it All Work?

# The MCP Stack

**Application Layer**
*(Your AI application (Claude Desktop, Cursor, etc.)*
*Examples: Claude Desktop, VS Code with MCP, Custom AI agents)*

**Protocol Layer**
*(MCP Protocol - defines message format, handshakes, tools, resources)*

**Transport Layer**
*(HOW messages are delivered between client and server)*

**Network Layer**
*(Physical network infrastructure (if remote) )*

# MCP Transports

An MCP Transport is the **communication** mechanism that carries MCP protocol messages between a client and server.

*The delivery system for MCP Messages*

# The MCP Stack

## Traditional Mail System

**Message:** The letter content (MCP Protocol)
**Envelope:** Address format (JSON-RPC)
**Delivery Method:** Postal service, email, courier (Transport)
**Infrastructure:** Roads, internet cables (Network)

## MCP Transport

**Message:** MCP protocol data
**Format:** JSON-RPC structure
**Delivery Method:** Stdio, SSE, HTTP (Transport)
**Infrastructure:** Process pipes, network connections

# MCP Transports

## MCP Transports

A transport handles the underlying mechanics of how messages are sent and received between the client and server.

1. For servers running locally: **stdio** (standard input output)

2. For remote servers:
   a. **HTTP+SSE (Server Sent Events)** (from protocol version 2024-11-05)
   b. **Streamable HTTP** (as of protocol version 2025-03-26)

# Key Transport Concepts

- Independent of MCP protocol content
- Same MCP server can use different transports
- Transport choice affects performance and capabilities
- Transport determines local vs remote server support
- Transport handles connection reliability and streaming
- Transport provides security and authentication layers

# Why Does Transport Matter?

- **Same protocol, Different Delivery:**
  - Messages can be delivered through different transport mechanisms (*just like sending a message via email, text, or postal mail)*
- **Types of Transport:**
  - Local
  - Remote
- **Transport trade Offs:**
  - Local vs remote
  - Speed
  - Real-time, complexity, compatibility

# The Three MCP Transports

| | |
|---|---|
| STDIO | Direct process-to-process communication (*local)* |
| SSE (Server-Sent Events) | Enables real-time streaming from server to client (like news feed) |
| Streamable HTTP | Enables streaming responses while maintaining HTTP’s universal compatibility. |

# The Three MCP Transports

STDIO

## 🔧 How It Works

- 🔗 **Client launches server as child process**
- 📤 **Messages sent via server's stdin**
- 📥 **Responses received from server's stdout**
- ⚡ **No network overhead - direct memory access**
- 🔒 **Inherently secure (local process isolation)**

# The Three MCP Transports

STDIO

## 🎯 Perfect Use Cases

### Desktop Applications

Claude Desktop connecting to local file servers, database tools, or system utilities. Maximum speed for local workflows.

### Development Tools

IDEs connecting to language servers, linters, or build tools. Fast feedback loops for coding assistance.

### Local Automation

Scripts and automation tools that need to process local files, run system commands, or access local databases.

### Prototyping

Quick prototypes and experiments where you want minimal setup complexity and maximum performance.

# The Three MCP Transports

STDIO

## ⚖️ Stdio Pros & Cons

### ✅ Advantages

- ✓ Fastest possible performance (no network layer)
- ✓ Zero network configuration required
- ✓ Automatic security (process isolation)
- ✓ Simple implementation
- ✓ No firewall or port issues
- ✓ Perfect for local development

### ❌ Limitations

- ✗ Local servers only (same machine)
- ✗ Can't connect to remote services
- ✗ Process lifecycle coupling
- ✗ No load balancing possible
- ✗ Single server instance per client
- ✗ Not suitable for web applications

# The Three MCP Transports

SSE (Server-Sent Events)

## 🔧 How It Works

- 🌐 **HTTP POST for client-to-server requests**
- 📡 **SSE stream for server-to-client responses**
- 🔄 **Automatic reconnection on connection loss**
- ⚡ **Real-time server push capabilities**
- 🌍 **Works across networks and browsers**

# The Three MCP Transports

SSE (Server-Sent Events)

## 🎯 Perfect Use Cases

🌐

### Web Applications

Browser-based AI interfaces that need real-time updates. Perfect for chat applications and live dashboards.

📊

### Live Monitoring

Real-time system monitoring, log streaming, or live data visualization where immediate updates are crucial.

💬

### Chat & Collaboration

Multi-user environments where servers need to push notifications, messages, or state changes to clients.

🎮

### Interactive Applications

Applications requiring server-initiated updates, like live tutorials, interactive demos, or gaming scenarios.

# The Three MCP Transports

SSE (Server-Sent Events)

## ⚖️ SSE Pros & Cons

### ✅ Advantages

- ✓ Real-time server push capabilities
- ✓ Automatic reconnection built-in
- ✓ Browser native support
- ✓ Works across networks and firewalls
- ✓ Efficient for streaming responses
- ✓ Lower latency than polling

### ❌ Limitations

- ✗ Requires HTTP server infrastructure
- ✗ More complex setup than Stdio
- ✗ Network dependency and latency
- ✗ Browser connection limits
- ✗ Requires handling connection state
- ✗ Not suitable for simple scripts

# The Three MCP Transports

Streamable HTTP

## 🔧 How It Works

- 🌐 **HTTP/HTTPS with chunked transfer encoding**
- 📊 **Streaming responses for large data/real-time output**
- 🔄 **Request-response pattern with streaming capability**
- ⚖️ **Works with load balancers and CDNs**
- 🏗️ **Integrates with existing web infrastructure**

**Built-in SSL/TLS security**


- ⚖️ **Works with load balancers and CDNs**
- 🏗️ **Integrates with existing web infrastructure**

# The Three MCP Transports

Streamable HTTP

## 🎯 Perfect Use Cases

🏢

### Enterprise Systems

Large-scale deployments requiring load balancing, monitoring, and integration with existing enterprise infrastructure.

☁️

### Cloud Services

Cloud-hosted MCP servers that need to serve multiple clients, handle variable loads, and integrate with cloud infrastructure.

🔌

### API Integration

When you need maximum compatibility with existing API infrastructure, monitoring tools, and security policies.

📱

### Mobile Applications

Mobile apps that need reliable, firewall-friendly communication with MCP servers across varying network conditions.

# The Three MCP Transports

Streamable HTTP

## ⚖️ HTTP Pros & Cons

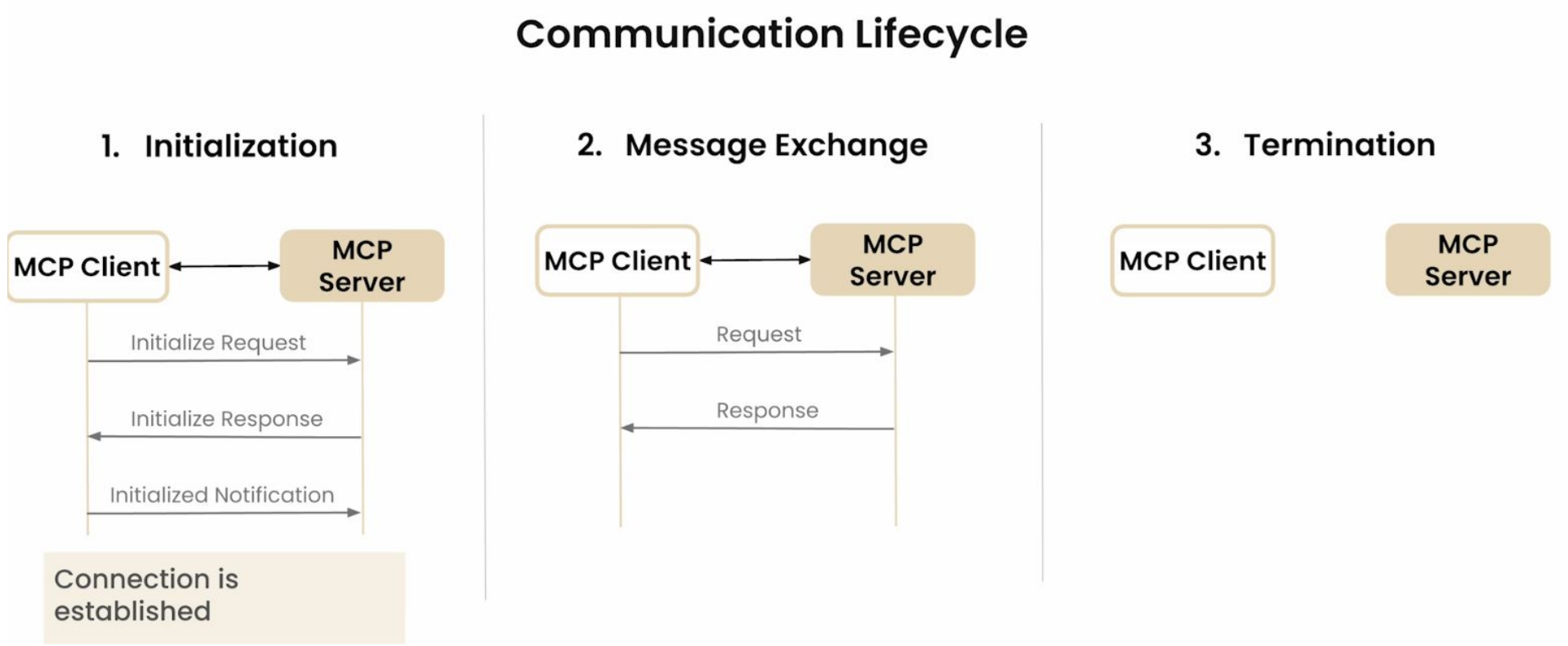
### ✅ Advantages

- ✓ Universal compatibility and standards
- ✓ Excellent scalability with load balancers
- ✓ Built-in security (HTTPS/TLS)
- ✓ Firewall and proxy friendly
- ✓ Extensive tooling and monitoring
- ✓ Caching and CDN support
- ✓ Works in any environment

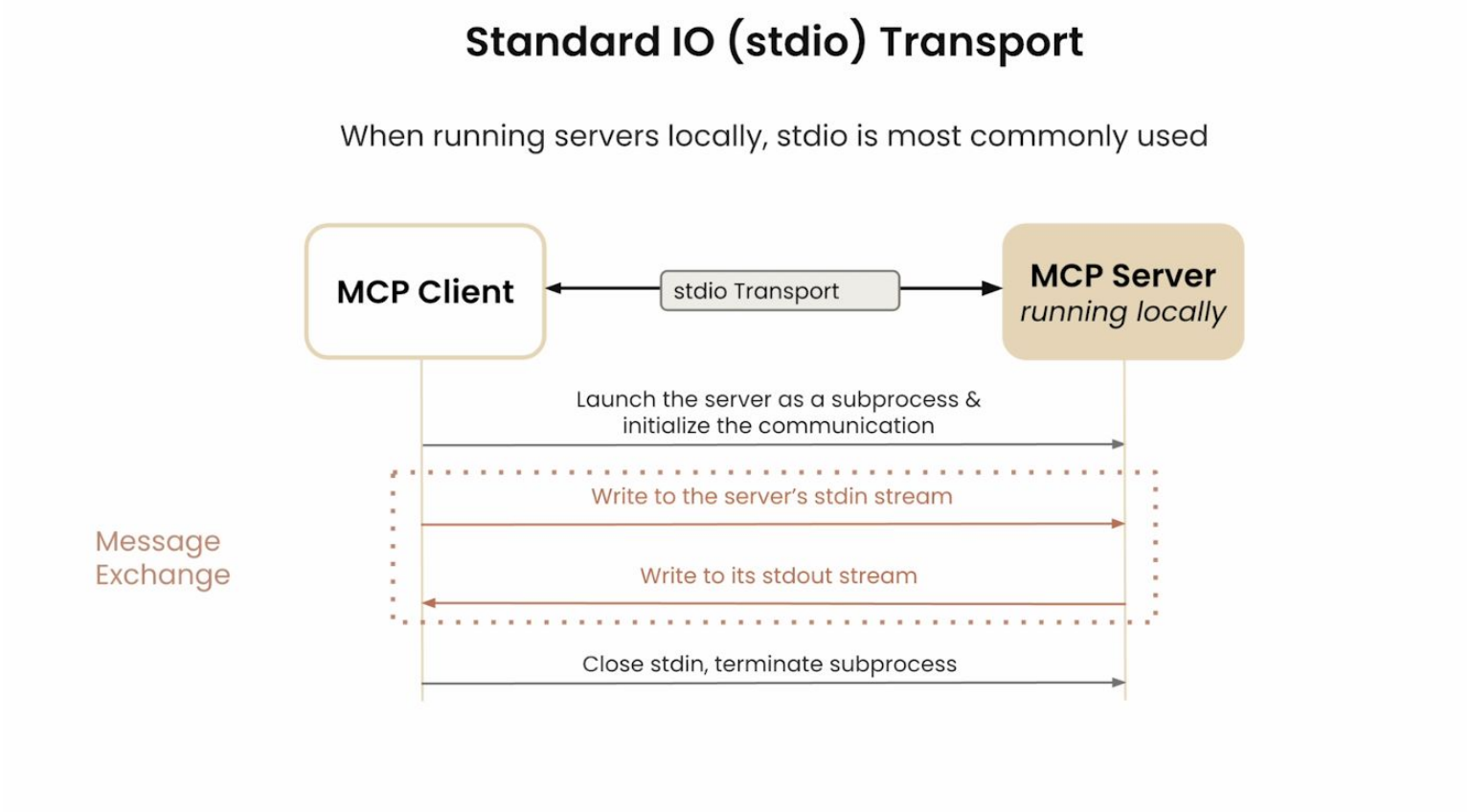
### ❌ Limitations

- ✗ Higher overhead than direct protocols
- ✗ Request-response pattern only
- ✗ No automatic server push
- ✗ More complex server setup
- ✗ Stateless (requires session management)
- ✗ Higher latency than local transports

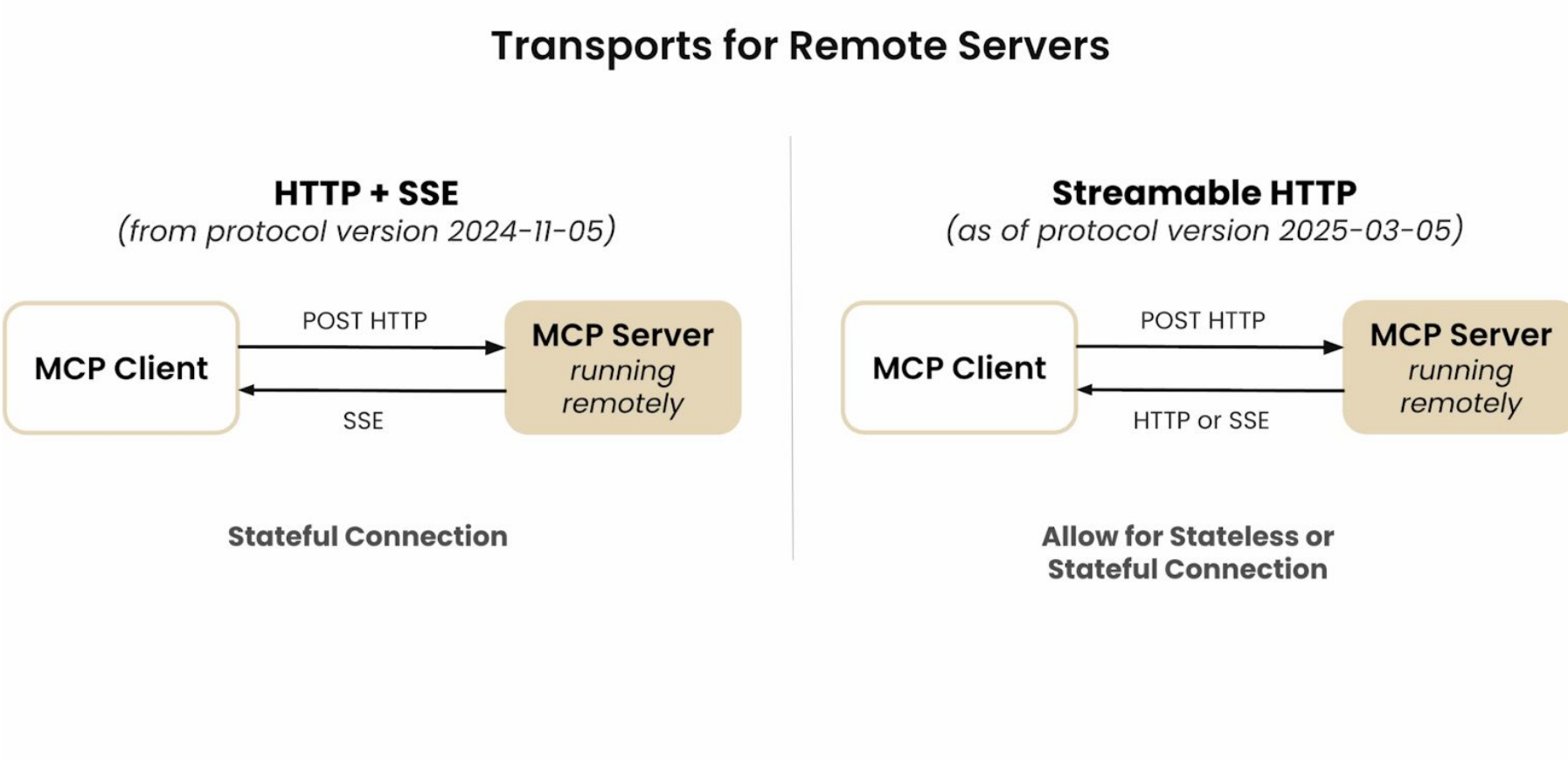
# Communication Lifecycle



# MCP Transports



# MCP Transports



# MCP - MCP Client Component

## Resources

- Static or dynamic data sources
- Real-time data updates
- Structured content delivery
- Searchable information repositories
- Live data feeds and monitoring

***Examples:*** *File contents, database records, API responses, live metrics, documentation*

## Tools

- Executable functions
- Parameterized actions
- Sandboxed execution
- Rich return data
- State management

***Examples:*** *File operations, API calls, database queries, code execution, calculations*

## Prompts

- Reusable prompt templates
- Variable interpolation
- Contextual prompting
- Conditional logic
- Prompt libraries

***Examples:*** *Code review templates, analysis prompts, specialized instructions*

# The Secure Conversation: HTTPS and JSON-RPC

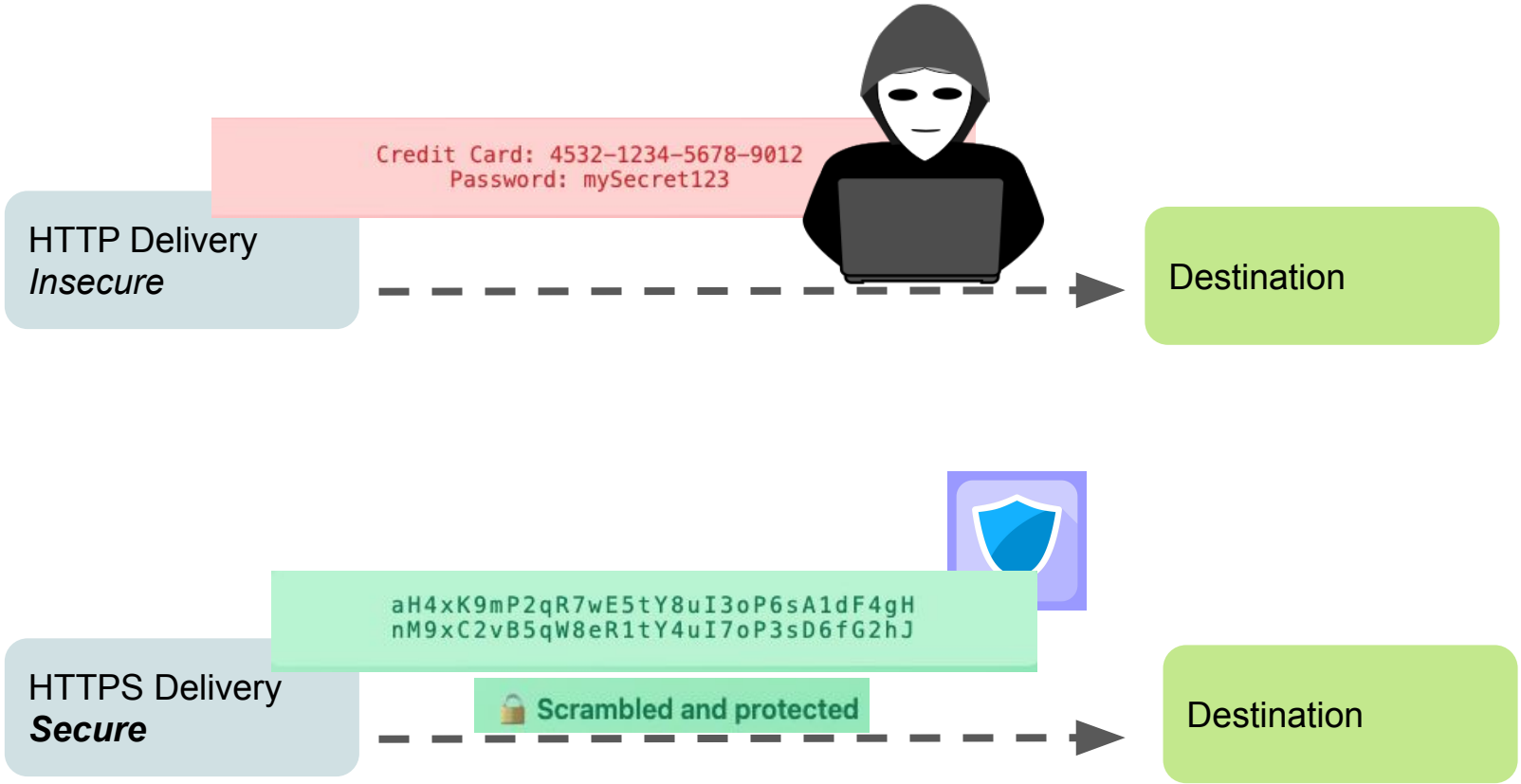
Communication

Security

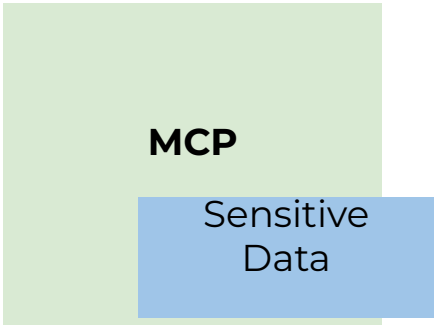
HTTP (basic protocol)

**HTTPS** (Hypertext Protocol Secure)

Transport Layer Security (TLS)


Credit Card: 4532-1234-5678-9012
Password: mySecret123
HTTP Delivery
Insecure
Destination
aH4xK9mP2qR7wE5tY8uI3oP6sA1dF4gH
nM9xC2vB5qW8eR1tY4uI7oP3sD6fG2hJ
Scrambled and protected
HTTPS Delivery
Secure
Destination

# Why HTTPS Matters for MCP




HTTPS ensures:

- All communication between MCP clients and servers is protected

**Encryption**
- *Data is scrambled*

**Authentication**
- *Verifies you're connecting to the legitimate server...*

**Integrity**
- *Ensures data hasn't been tampered with during transit*

**Protection**
- *Shields sensitive MCP ops from malicious actors*

# Why MCP - What is it?

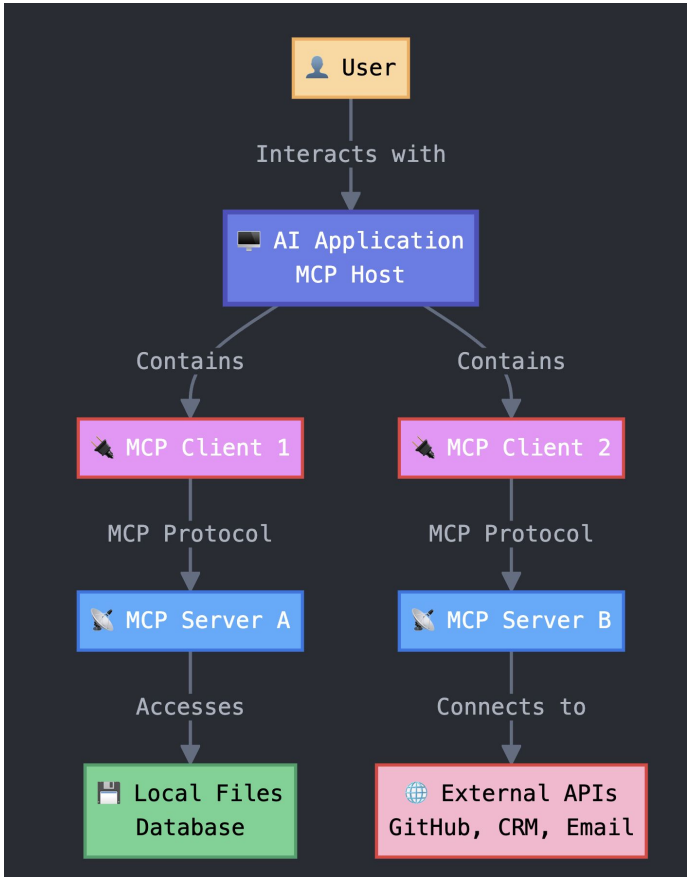
**REST APIs**
*(standardized how web applications interact with the backend)*

**LSP**
*(standardized how IDEs interact with language-specific tools)*

**MCP**
*(standardized how AI applications interact with external systems)*

- MCP is an open protocol that standardizes how LLM applications connect to and work with external tools and data sources.

# The Architecture: A Collaborative Ecosystem



# Conclusion/Summary

## Before MCP

- 📚 Static knowledge only
- 💬 Intelligent text generators
- 🔒 Isolated from external data
- ⚡ Limited to training data
- 🛠️ No external tool access

MCP Enables

## With MCP

- 🔄 Dynamic, current information
- 🎯 Autonomous agentic systems
- 🌐 Connected to external world
- 👁️ Perceive, reason, and act
- 🔧 Access to tools and data

# Conclusion/Summary

### Standardized

One universal protocol for all AI-to-system connections

### Secure

Built-in security measures protect sensitive data and operations

### Flexible

Adaptable bridge between LLMs and any external system

# MCP Server Development!

## Hands-on

# Dev Tools You'll need

1. VS Code
   a. Free and the best IDE (also serves as MCP Client for testing…)
   b. Install Python
   c. Claude Desktop (an easy MCP Host with MCP Client to use)
   d. Install UV for python dependency management
   e. Install npm as well
   *f. OpenAI API key - optional in our case*

# Hands-on - Building MCP Servers

# Using Prebuilt MCP Servers

Hands on: Using prebuilt MCP servers

# Hands-on - Build Your First MCP Server

# Hands-on - Build Your First MCP Client

# Hands-on - Build Your First MCP Client

# MCP Core Concepts (Continued)

- Have a portfolio
- At least 2 big, fairly complex projects
- Have a Github account with a few projects

# Build a Chatbot that Uses Tools

# MCP Server - Deep Dive

# MCP Resources and Prompts - Deep Dive

These are the other two main MCP Primitives:

- Prompts
- Resources

**The main focus is generally the primitive Tools because it's the most Used and useful!*

# MCP Primitives

## Resources

- Static or dynamic data sources
- Real-time data updates
- Structured content delivery
- Searchable information repositories
- Live data feeds and monitoring

***Examples:*** *File contents, database records, API responses, live metrics, documentation*

## Tools

- Executable functions
- Parameterized actions
- Sandboxed execution
- Rich return data
- State management

***Examples:*** *File operations, API calls, database queries, code execution, calculations*

## Prompts

- Reusable prompt templates
- Variable interpolation
- Contextual prompting
- Conditional logic
- Prompt libraries

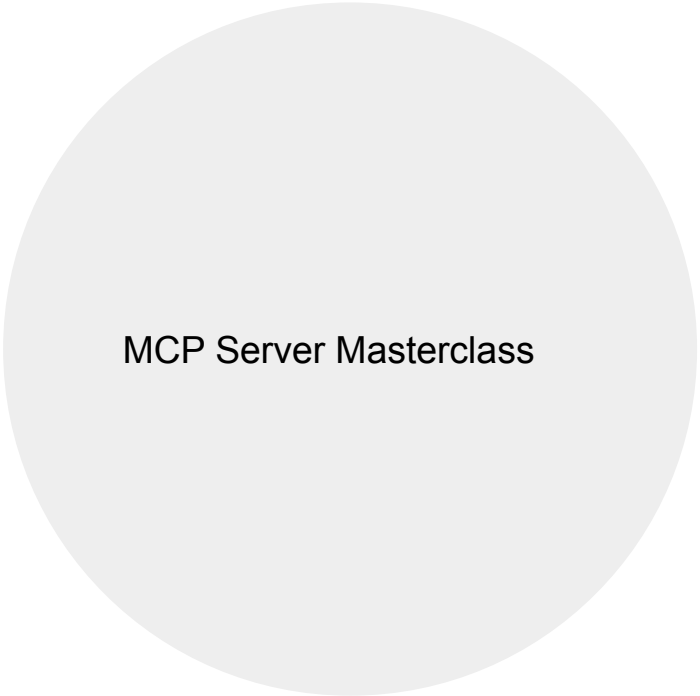
***Examples:*** *Code review templates, analysis prompts, specialized instructions*

# Deploying and Publishing MCP Servers

# STDIO vs Streamable HTTP

**Congratulations!!!!!!!!!**

# Wrap up - Next Steps




- Practice by building MCP servers.
  - Keep learning about MCP servers
  - MCP updates/news
  - https://github.com/modelcontextprotocol
  - https://modelcontextprotocol.io/docs/getting-started/intro
- This is the beginning of your MCP server journey.  Keep going!

# Build a full-fledged MCP Server

READ thisL:
https://support.anthropic.com/en/articles/10949351-getting-started-with-local-mcp-servers-on-claude-desktop

For difference between remote mcp and local mcps... and integrations!