MCP & A2A

Understanding MCP’s role in AI model interoperability

Slides & Resources

Naveen Gopalakrishna

AI Apps Solution Engineer

Why Model Context Protocol

Challenges in Model interoperability & Model Communication Frameworks

Siloed AI Environments

AI models frequently operate independently, creating isolated systems that cannot effectively communicate or share data.

Context Sharing Difficulties

Sharing contextual information between models is complex, often leading to misinterpretations and reduced accuracy.

Limits to Collaborative Workflows

Challenges in interoperability hinder seamless collaboration, limiting overall AI model performance and efficiency.

Lack of Flexibility

Traditional models often cannot adapt to changing AI ecosystem requirements, limiting dynamic communication capabilities.

Scalability Challenges

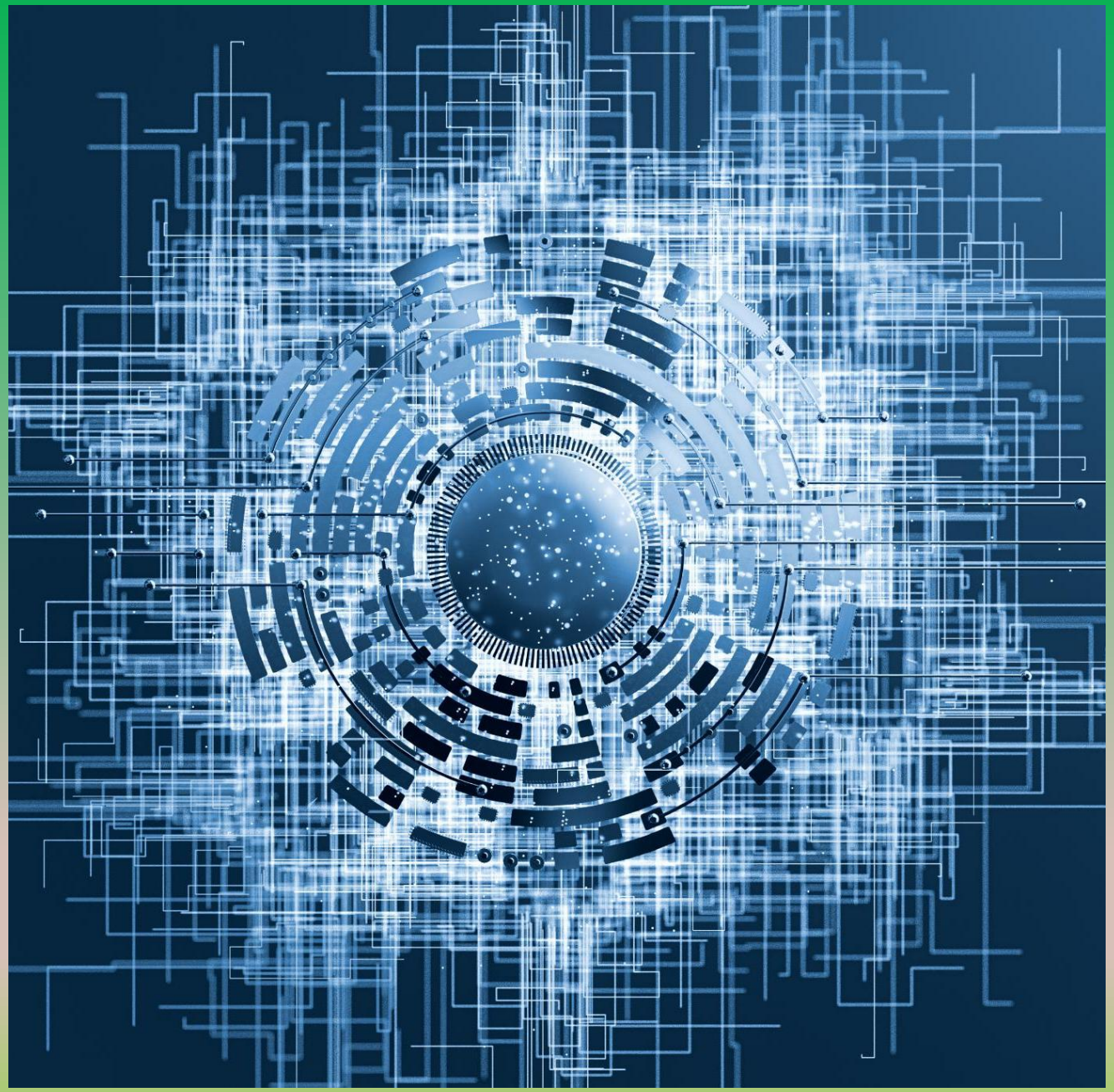
These frameworks struggle to scale efficiently with increasing data and model complexity in AI environments.

Poor Context Continuity

Traditional frameworks fail to maintain continuous context, hindering complex interaction across different AI models.

Lack of Standardization

Absence of unified standards restricts interoperability and integration across diverse AI communication frameworks.



Opportunities Unlocked by MCP for Scalable AI Systems

Efficient Context Exchange

MCP protocols facilitate seamless and efficient exchange of context between AI models for improved collaboration.

Interoperability Across Models

MCP enables different AI models to work together regardless of architecture or framework differences.

Extensibility for Scalability

Protocols designed by MCP support scalable AI systems that grow with increasing collaboration and data needs.

When to use Model Context Protocol vs Agent tools

Real-World MCP & Agent Tool Uses

Use Agent Tools for:

- Simple web scraping
- Basic math operations
- Single-use API integrations
- Agent-specific utilities
- Quick data transformations

Build MCP Server for:

- Salesforce integration (multiple teams need access)
- Corporate document management system
- Real-time monitoring dashboards
- Custom ERP systems
- Multi-tenant SaaS platforms

Hybrid Approach:

```
MCP Servers for:
├── Core business data (CRM, ERP, databases)
├── Shared infrastructure (file systems, monitoring)
└── Complex integrations (legacy systems, real-time data)

Agent Tools for:
├── Agent-specific workflows
├── Simple external APIs
└── Rapid prototyping features
```

The key principle: **If multiple AI applications or agents will need the same data/functionality, build an MCP server. If it's specific to one agent's workflow, use dedicated tools.**

Decision Matrix

Factor	MCP Server	Agent Tools
Multiple AI apps need access	✅ Perfect fit	❌ Requires duplication
Complex data sources	✅ Standardized handling	❌ Each agent reinvents
Real-time updates needed	✅ Built-in subscriptions	❌ Complex to implement
Enterprise security requirements	✅ Centralized control	❌ Distributed security
Simple API calls	❌ Overkill	✅ Perfect fit
Rapid prototyping	❌ More setup overhead	✅ Quick and dirty
Legacy system integration	✅ Abstraction layer	❌ Too complex

What Is Model Context Protocol

Definition and Overview of MCP

Standardized Protocol

MCP provides a common framework for exchanging contextual information among diverse AI systems.

Facilitation of Information Sharing

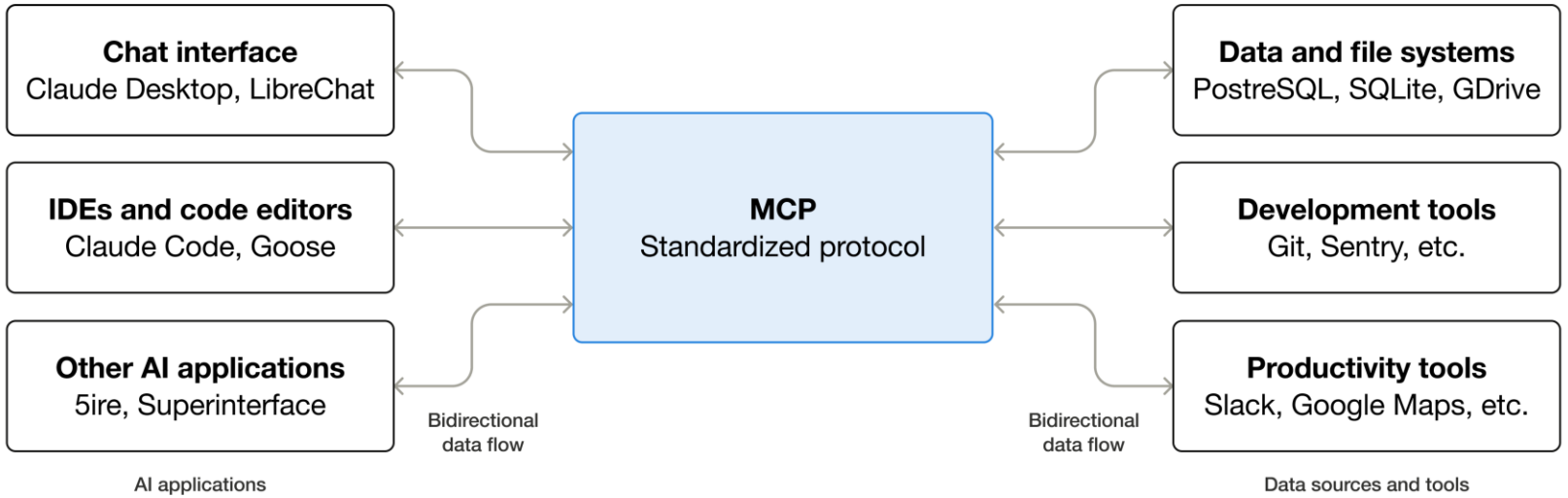
MCP enables seamless data sharing that improves collaboration and integration between AI models.

Enhanced AI Cooperation

MCP supports improved cooperation leading to more powerful and capable AI model ecosystems.

What is the Model Context Protocol (MCP)?

MCP is an open protocol that enables seamless integration between **LLM applications** and your **tools & data sources**.



What is the Model Context Protocol (MCP)?

Using MCP, AI Applications like Copilot, GHCP, ChatGPT can connect to data sources(e.g. local files, databases), tools(e.g. search engines, calculators) and workflows (e.g. specialized prompts)—enabling them to access key information and perform tasks.

USB C for AI Applications

APIs

Standardize how **web applications** interact with the **backend**:

- Servers
- Databases
- Services

LSP

Standardizes how **IDEs** interact with **language-specific tools**:

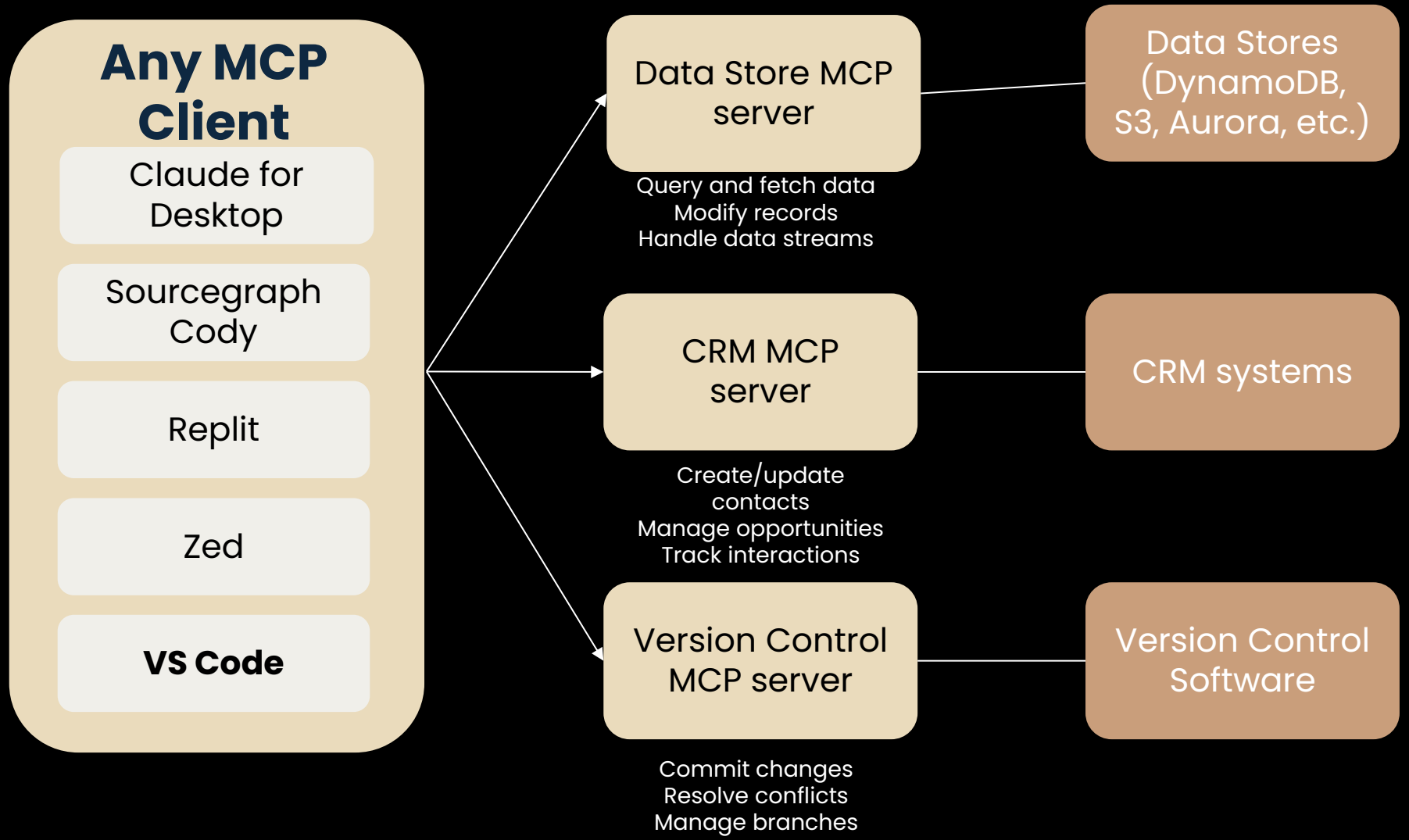
- Code navigation
- Code analysis
- Code intelligence

MCP

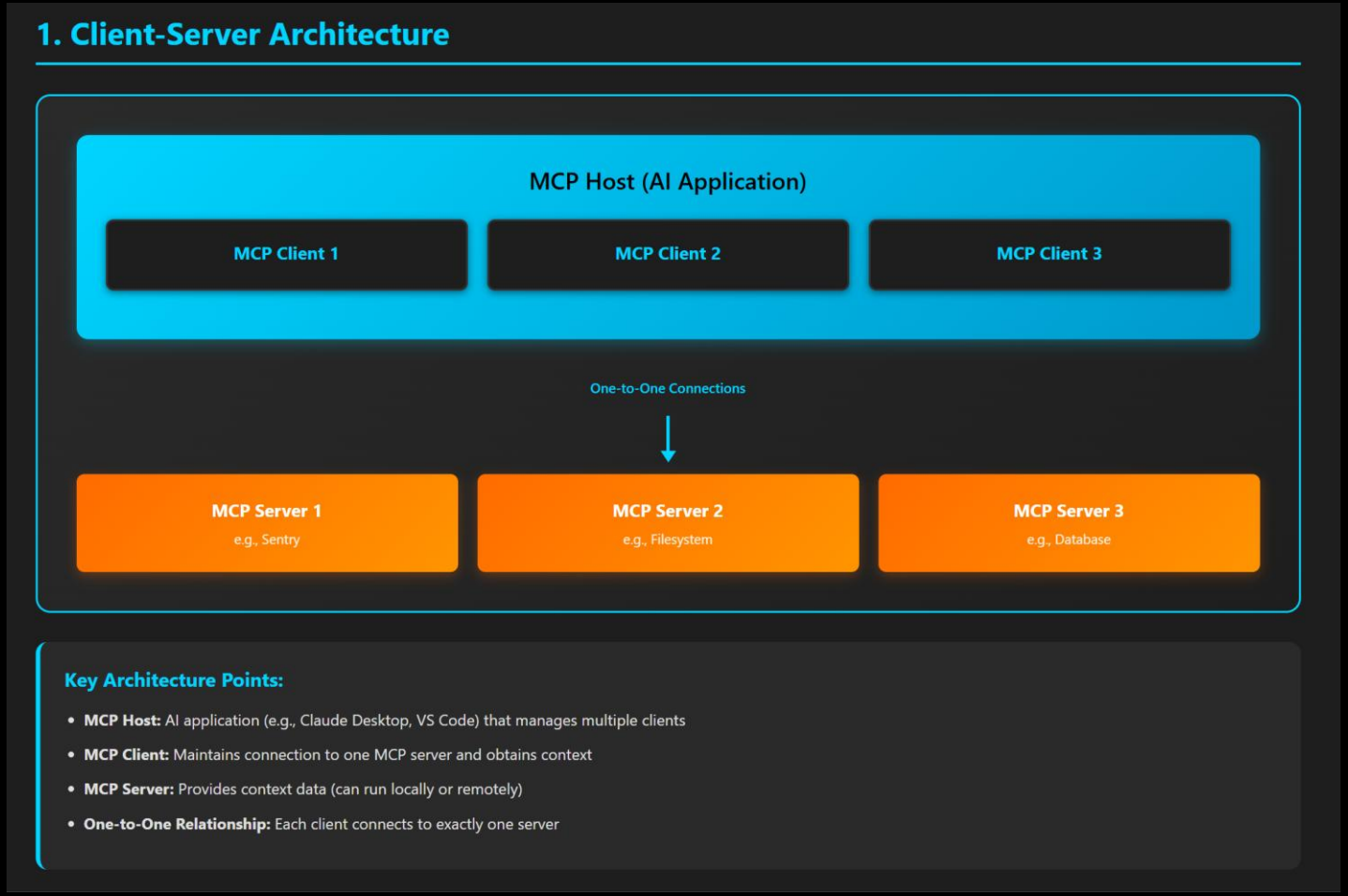
Standardizes how **AI applications** interact with **external systems**:

- Prompts
- Tools
- Data & resources
- Sampling

MCP: Standardized AI Development



Architecture of Model Context Protocol



MCP Architecture

2. Protocol Layers

Data Layer

JSON-RPC 2.0 Protocol

- Lifecycle Management
- Tools, Resources & Prompts
- Client Features (Sampling, Elicitation)
- Notifications & Progress Tracking

Transport Layer

Communication Mechanisms

- **Stdio:** Local process communication
- **HTTP:** Remote server communication with SSE
- Connection establishment
- Message framing & authentication

MCP Architecture

3. Core Primitives (Server Features)

🛠️ Tools

Executable functions that AI can invoke to perform actions like file operations, API calls, and database queries

📁 Resources

Data sources that provide contextual information such as file contents, database records, and API responses

💬 Prompts

Reusable templates that structure LLM interactions including system prompts and few-shot examples

Primitive Operations:

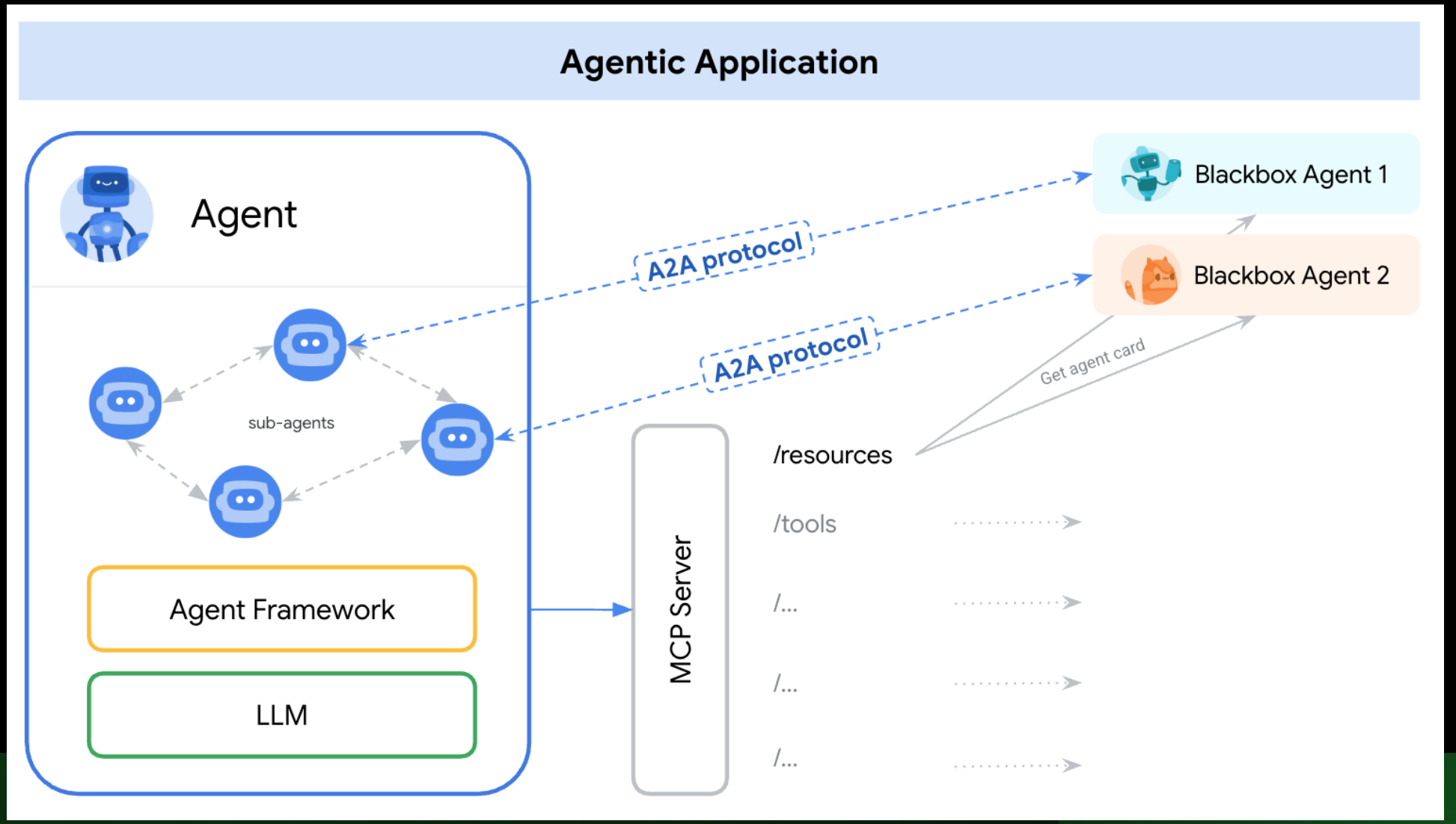
- **Discovery:** `*/list` methods to discover available primitives
- **Retrieval:** `*/get` methods to retrieve specific primitives
- **Execution:** `tools/call` method to execute tools
- **Dynamic Listings:** Primitives can change during runtime

MCP Architecture

4. Communication Flow

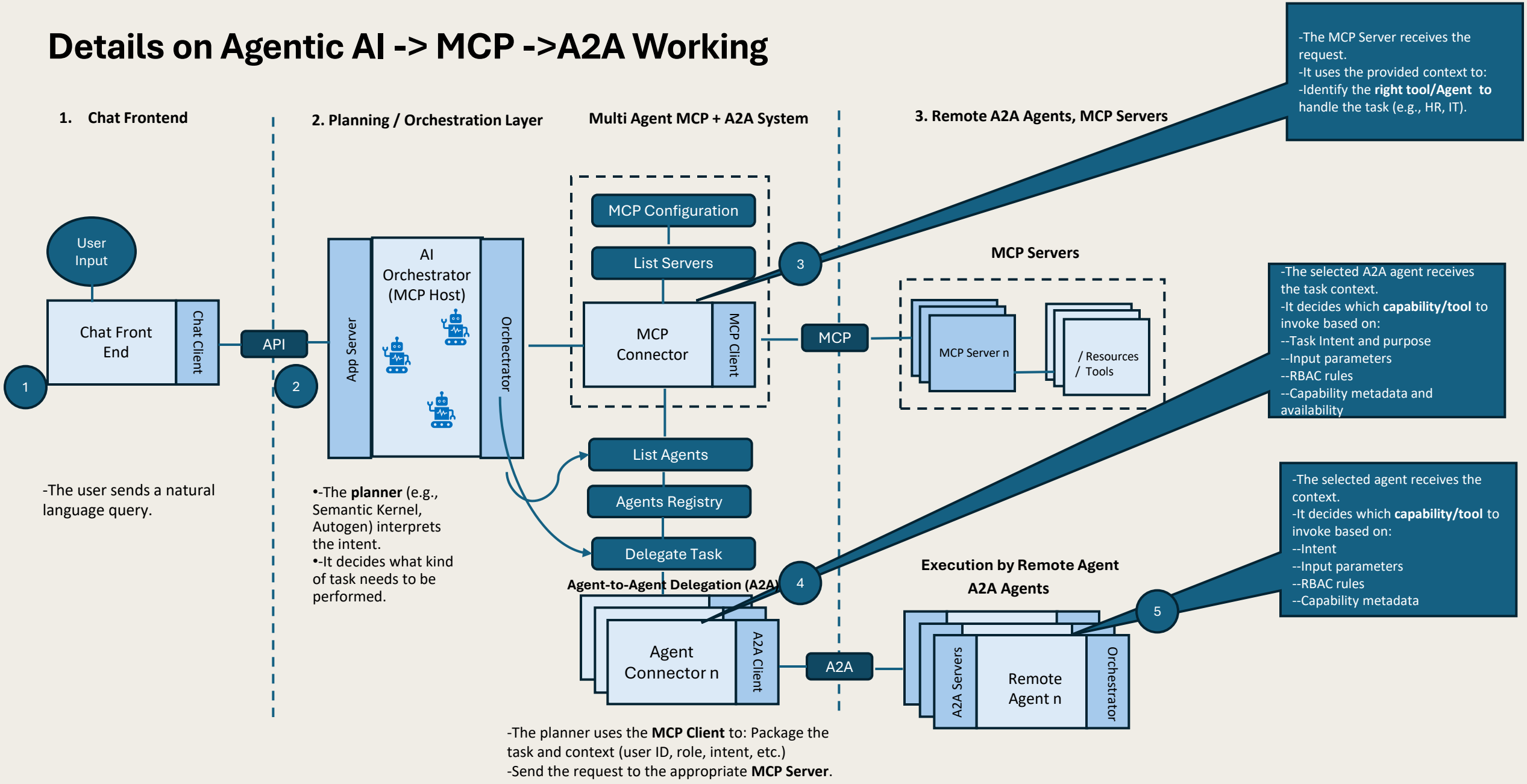
- **Initialization:** Client and server negotiate capabilities via handshake
- **Discovery:** Client lists available tools/resources/prompts
- **Execution:** Client calls tools with arguments, receives structured responses
- **Notifications:** Server sends real-time updates when capabilities change
- **Stateful Protocol:** Maintains connection state throughout session

MCP Architecture



MCP Workflow

Details on Agentic AI -> MCP ->A2A Working



MCP Inspector

Interactive Developer Tool for Testing & Debugging

The MCP Inspector is an interactive developer tool designed for testing and debugging Model Context Protocol servers. It provides a comprehensive interface to test all aspects of your MCP server implementation.

🚀 Getting Started

Basic Usage:

```
npx @modelcontextprotocol/inspector <command>
```

📦 NPM Package:

```
npx -y @modelcontextprotocol/inspector npx
@modelcontextprotocol/server-filesystem /path/to/directory
```

PyPi Package:

```
npx @modelcontextprotocol/inspector uvx mcp-server-git --repository
~/code/repo.git
```

Local TypeScript:

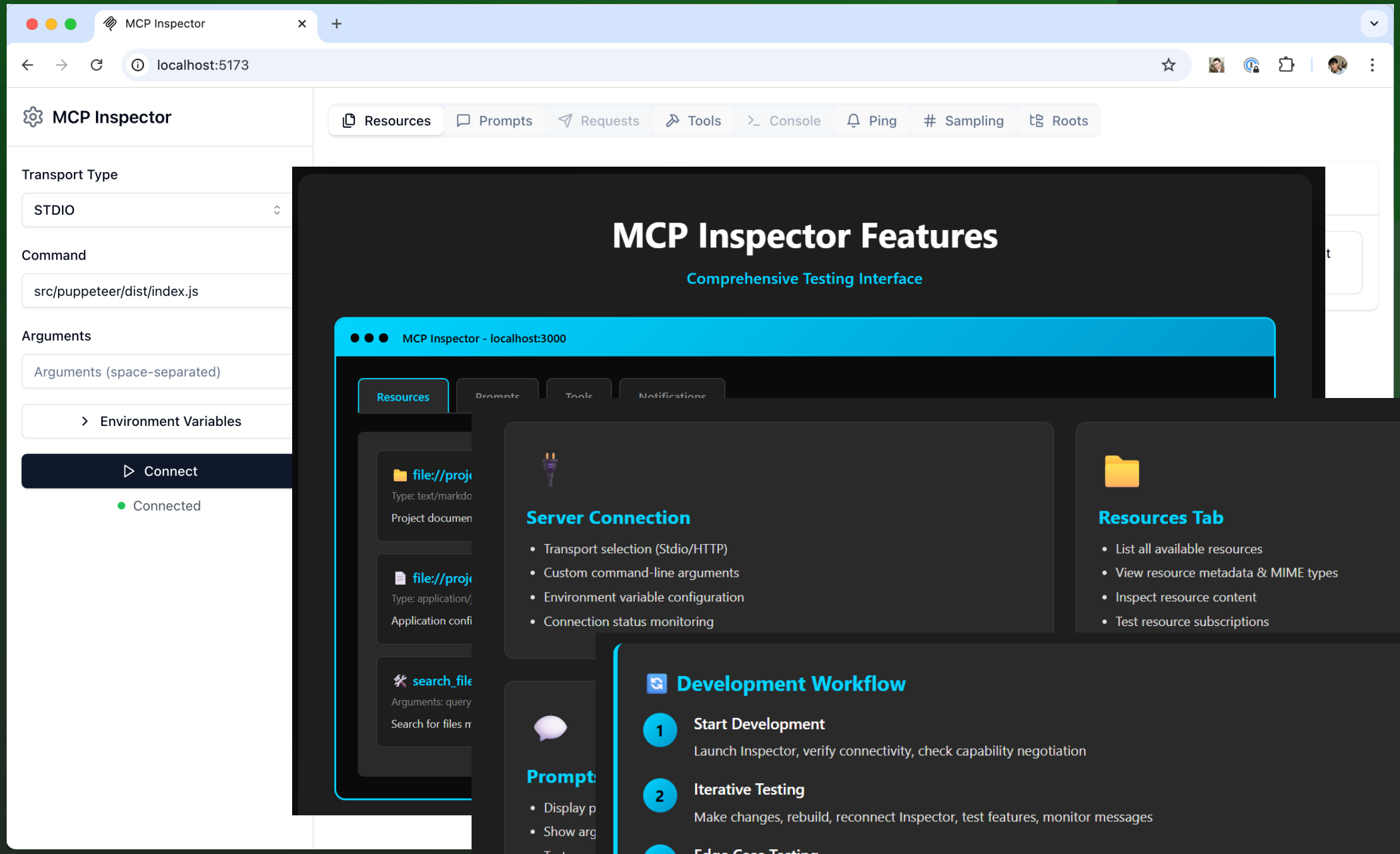
```
npx @modelcontextprotocol/inspector node path/to/server/index.js
```

Local Python:

```
npx @modelcontextprotocol/inspector uv --directory path/to/server
run package-name
```

💡 No Installation Required!

The Inspector runs directly through npx without requiring any installation. Simply run the command and start testing your MCP server immediately.


MCP Inspector
localhost:5173
MCP Inspector
Resources
Prompts
Requests
Tools
Console
Ping
Sampling
Roots
Transport Type
STDIO
Command
src/puppeteer/dist/index.js
Arguments
Arguments (space-separated)
Environment Variables
Connect
Connected
MCP Inspector Features
Comprehensive Testing Interface
MCP Inspector - localhost:3000
Resources
Server Connection
Transport selection (Stdio/HTTP)
Custom command-line arguments
Environment variable configuration
Connection status monitoring
Resources Tab
List all available resources
View resource metadata & MIME types
Inspect resource content
Test resource subscriptions
Development Workflow
1
Start Development
Launch Inspector, verify connectivity, check capability negotiation
2
Iterative Testing
Make changes, rebuild, reconnect Inspector, test features, monitor messages
3
Edge Case Testing
Test invalid inputs, missing arguments, concurrent operations, error handling

MCP Center

Search MCP servers...

Build Your Scalable, Enterprise-Ready MCP Registry with Azure API Center

Filter by

Types

- Local
- Remote

Vendors

- Microsoft
- Partner

Endpoint

- Public

Displaying **24** items

Sort by

MCP | REMOTE
Vercel
With Vercel MCP, you can explore projects, inspect failed deployments, fetch logs, and more right from your AI client.

MCP | LOCAL
Tavily MCP
The Tavily MCP server offers a suite of tools, including search, extract, map, and crawl.

MCP | REMOTE
Stripe
Payment processing and financial infrastructure tools.

MCP | REMOTE
Postman
Postman's remote MCP server connects AI

MCP | LOCAL
Playwright
This server enables LLMs to interact with web

MCP | REMOTE
Pipedream
Securely connect to 10,000+ tools from 3,000+

Connect models to the real world

Servers and tools from the community that connect models to files, APIs, databases, and more.

Search MCPs

All MCP servers 44

Markitdown

Install

Convert various file formats (PDF, Word, Excel, images, audio) to Markdown.

By microsoft ☆ 80,749

Context7

Install

Get up-to-date, version-specific documentation and code examples from any library or framework.

By upstash ☆ 33,136

GitHub

Install

Official GitHub MCP Server that connects AI tools directly to GitHub's platform. Enables AI agents to read repositories,...

By github ☆ 23,389

skd-apicenter-mcp | APIs

API Center

Search

Access control (IAM)
Tags
Resource visualizer
Events
Assets
Metadata
APIs
Environments
Dependency tracker (preview)
Discovery
MCP (preview)
Platforms
Integrations

Register an API | Refresh

Seamlessly register new APIs to your Azure API center, expanding your API inventory for enhanced transparency and oversight. Learn how
registration.

Search | API type : All | Lifecycle : All | Add filter

API title	API type	Lifecycle	Summary
Swagger Petstore	REST	Testing	A sample API that us
apimmcppet	MCP	Development	
aoai-gpt4o	REST	Development	
aoai-lb	REST	Development	
llama70b	REST	Development	
mslearnmcp	MCP	Development	
Swagger Petstore - OpenAPI 3.0	REST	Development	
skdgemini	REST	Development	

Microsoft MCP Resources

Model Context Protocol

Official MCP Catalog

https://github.com/microsoft/mcp

MCP for Beginners

https://github.com/microsoft/mcp-for-beginners

https://github.com/microsoft/mcp

microsoft / mcp

Type / to search

Code | Issues 141 | Pull requests 28 | Actions | Projects 2 | Models | Security

README | Code of conduct | Contributing | MIT license | Security

mcp Public

Microsoft MCP Servers

What is MCP?

Model Context Protocol (MCP) is an open protocol that standardizes how applications provide context to large language models (LLMs). It allows AI applications to connect with various data sources and tools in a consistent manner, enhancing their capabilities and flexibility. MCP follows a client-server architecture:

- **MCP Hosts**: Applications like AI assistants or IDEs that initiate connections.
- **MCP Clients**: Connectors within the host application that maintain 1:1 connections with servers.
- **MCP Servers**: Services that provide context and capabilities through the standardized MCP.

For more details, visit the official MCP website.

Which MCP Servers are built from this repository?

This repository contains core libraries, test frameworks, engineering systems, pipelines, and tooling for Microsoft M
Server contributors to unify engineering investments; and reduce duplication and divergence:

MCP Server	README	Source Code	CHANGELOG	Releases	Documentation	Troubleshooting	Supp
Azure MCP	Azure MCP README	Azure MCP Source Code	Azure MCP CHANGELOG	Azure MCP Releases	Azure MCP Documentation	Azure MCP Troubleshooting	Azu MCI Supp
Microsoft Fabric MCP	Fabric MCP README	Fabric MCP Source Code	Fabric MCP CHANGELOG	Fabric MCP Releases	Fabric Documentation	Fabric MCP Troubleshooting	Fabr MCI Supp

Which MCP Servers are available from Microsoft?

Azure

- **REPOSITORY**: microsoft/mcp
- **DESCRIPTION**: All Azure MCP tools in a single server. The Azure MCP Server implements the MCP specification to create a seamless connection between AI agents and Azure services. Azure MCP Server can be used alone or with the GitHub Copilot for Azure extension in VS Code.
- **CATEGORY**: `CLOUD AND INFRASTRUCTURE`
- **TYPE**: `Local`
- **INSTALL**: VS Code | VS Code Insiders | Visual Studio | IntelliJ IDEA

Azure AI Foundry

- **REPOSITORY**: azure-ai-foundry/mcp-foundry
- **DESCRIPTION**: A Model Context Protocol server for Azure AI Foundry, providing a unified set of tools for models, knowledge, evaluation, and more.
- **CATEGORY**: `CLOUD AND INFRASTRUCTURE`
- **TYPE**: `Local`
- **INSTALL**: VS Code | VS Code Insiders | Visual Studio

Azure DevOps

- **REPOSITORY**: Azure DevOps MCP Server - Public Preview
- **DESCRIPTION**: This TypeScript project provides a local MCP server for Azure DevOps, enabling you to perform a wide range of Azure DevOps tasks directly from your code editor.
- **CATEGORY**: `DEVELOPER TOOLS`
- **TYPE**: `Local`
- **INSTALL**: VS Code | VS Code Insiders | Visual Studio

Azure Kubernetes Service (AKS)

- **REPOSITORY**: Azure/aks-mcp
- **DESCRIPTION**: An MCP server that enables AI assistants to interact with Azure Kubernetes Service (AKS) clusters. It serves as a bridge between AI tools and AKS, translating natural language requests into AKS operations and returning the results in a format the AI tools can understand.

https://github.com/microsoft/mcp-for-beginners

microsoft / mcp-for-beginners

Type /

Code | Issues | Pull requests 2 | Actions | Projects | Models | Security | Insights

mcp-for-beginners Public

Watch 104

main | Branches | Tags | Go to file t | Add file | Code

leestott Merge pull request #451 from microsoft/update-translations 122b47a · 3 days ago 1,161 Commits

.devcontainer	Removed extra line!	last month
.github	Update co-op-translator.yml	last month
00-Introduction	Update 00-Introduction/README.md	3 weeks ago
01-C		
02-S		
03-G		
04-P		
05-A		
06-C		
07-L		
08-B		
09-C		
10-S		
11-M		
imag		

contributors 46 | issues 0 open | pull requests 2 open | PRs welcome

Watch 104 | Fork 3.7k | Star 12k

🚀 Model Context Protocol (MCP) Curriculum for Beginners

Learn MCP with Hands-on Code Examples in C#, Java, JavaScript, Rust, Python, and TypeScript

🧠 Overview of the Model Context Protocol Curriculum

Welcome to your jou
communicate with c
developers build int

Think of MCP as a u
computer, MCP lets
chatbot or working
flexible applications.

This curriculum is de
already understand
language. Every step

By the time you com
popular AI platforms
exciting adventure t

Official Docume

These resources hec
ript-applicati
l-context-pro
lcontextprot
ecurity

adme
T license
de of condi
curity polic
tivity
stom properties
.2k stars
4 watching
7k forks
repository

On Demand Content

MCP Dev Days July 2025

➡️ Watch on Demand - MCP Dev Days

Get ready for two days of deep technical insight, community connection, and hands-on learning at MCP Dev Days, a virtual event dedicated to the Model Context Protocol (MCP) — the emerging standard that bridges AI models and the tools they rely on. You can watch MCP Dev Days by registering on our event page: https://aka.ms/mcpdevdays.

Day 1: MCP Productivity, DevTools, & Community:

Is all about empowering developers to use MCP in their developer workflow and celebrating the amazing MCP community. We'll be joined with community members and partners such as Arcade, Block, Okta, and Neon to see how they are collaborating with Microsoft to shape an open, extensible MCP ecosystem. Real-world demos across VS Code, Visual Studio, GitHub Copilot, and popular community tools Practical, context-driven dev workflows Community-led sessions and insights Whether you're just getting started with MCP or already building with it, Day 1 will set the stage with inspiration and actionable takeaways.

Day 2: Build MCP Servers with Confidence

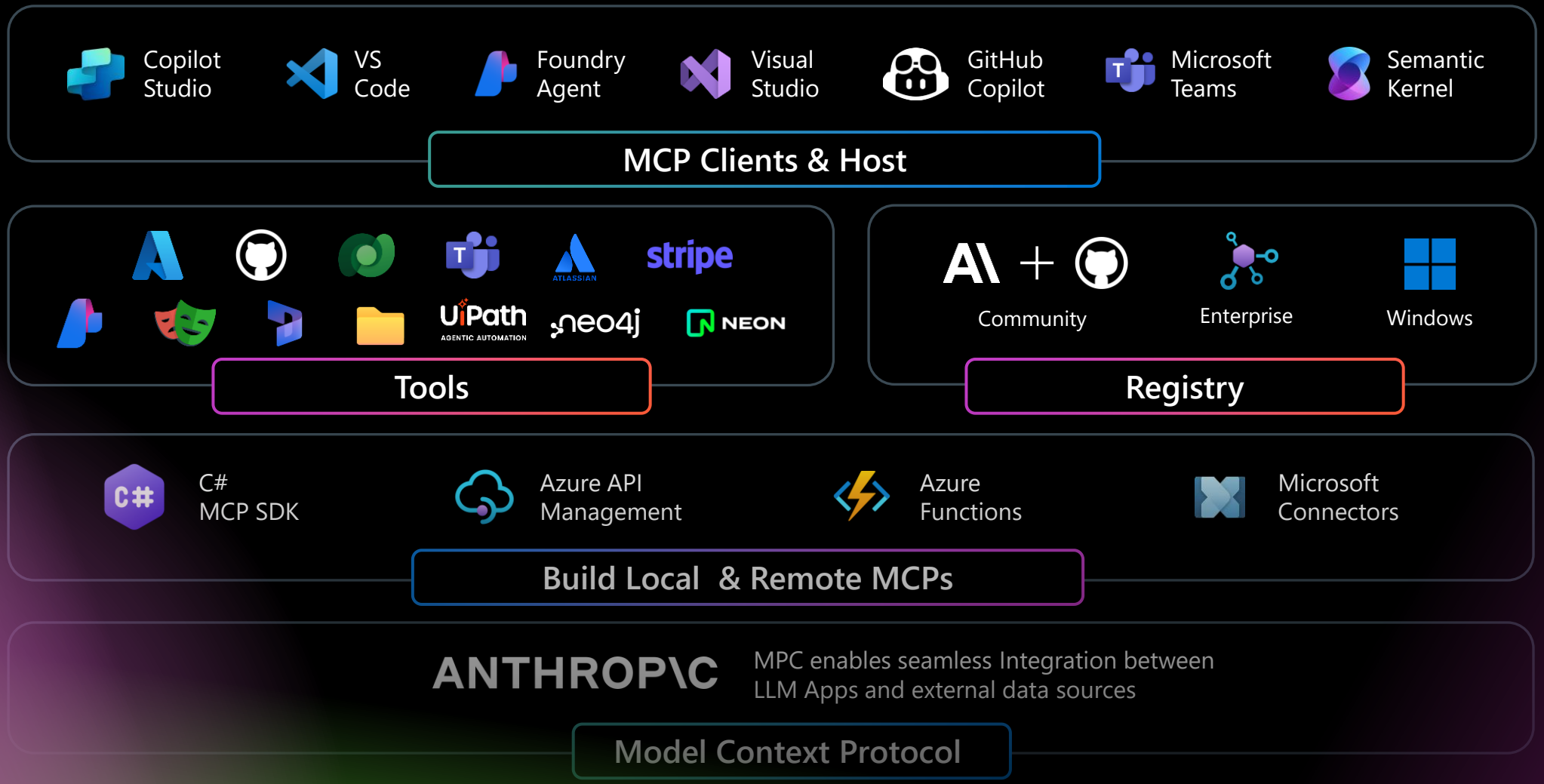
Is for MCP builders. We'll go deep into implementation strategies and best practices for creating MCP servers and integrating MCP into your AI workflows.

Topics include:

- Building MCP Servers and integrating them into agent experiences
- Prompt-driven development
- Security best practices
- Using building blocks like Functions, ACA, and API Management
- Registry alignment and tooling (1P + 3P)

If you're a developer, tool builder, or AI product strategist, this day is packed with the insights you need to build scalable, secure, and future-ready MCP solutions.

Model Context Protocol at Microsoft



Deployment Options

🚀 Serverless Functions

Deploy lightweight MCP servers as serverless functions for automatic scaling and cost-effective operation with zero infrastructure management.

📦 Container Services

Use services like **Azure Container Apps**, **AWS ECS**, or **Google Cloud Run** for containerized deployments with enhanced control and flexibility.

☸️ Kubernetes

Deploy and manage MCP servers in Kubernetes clusters for **high availability**, advanced orchestration, and enterprise-grade reliability.

🔌 API Management (APIM)

Leverage **API Management platforms** to expose MCP servers with built-in security, rate limiting, analytics, and centralized gateway management.

When to Use Each Option

🚀 Serverless Functions

Best for: Low-traffic, sporadic workloads, rapid prototyping, and cost optimization with minimal maintenance overhead.

📦 Container Services

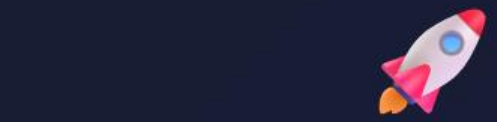
Best for: Moderate traffic, need for custom dependencies, microservices architecture, and balanced control with managed infrastructure.

☸️ Kubernetes

Best for: High-traffic production environments, complex multi-service deployments, enterprise requirements, and maximum control over orchestration.

🔌 API Management

Best for: Public-facing APIs, multi-tenant scenarios, enterprise security requirements, and centralized monitoring and analytics.

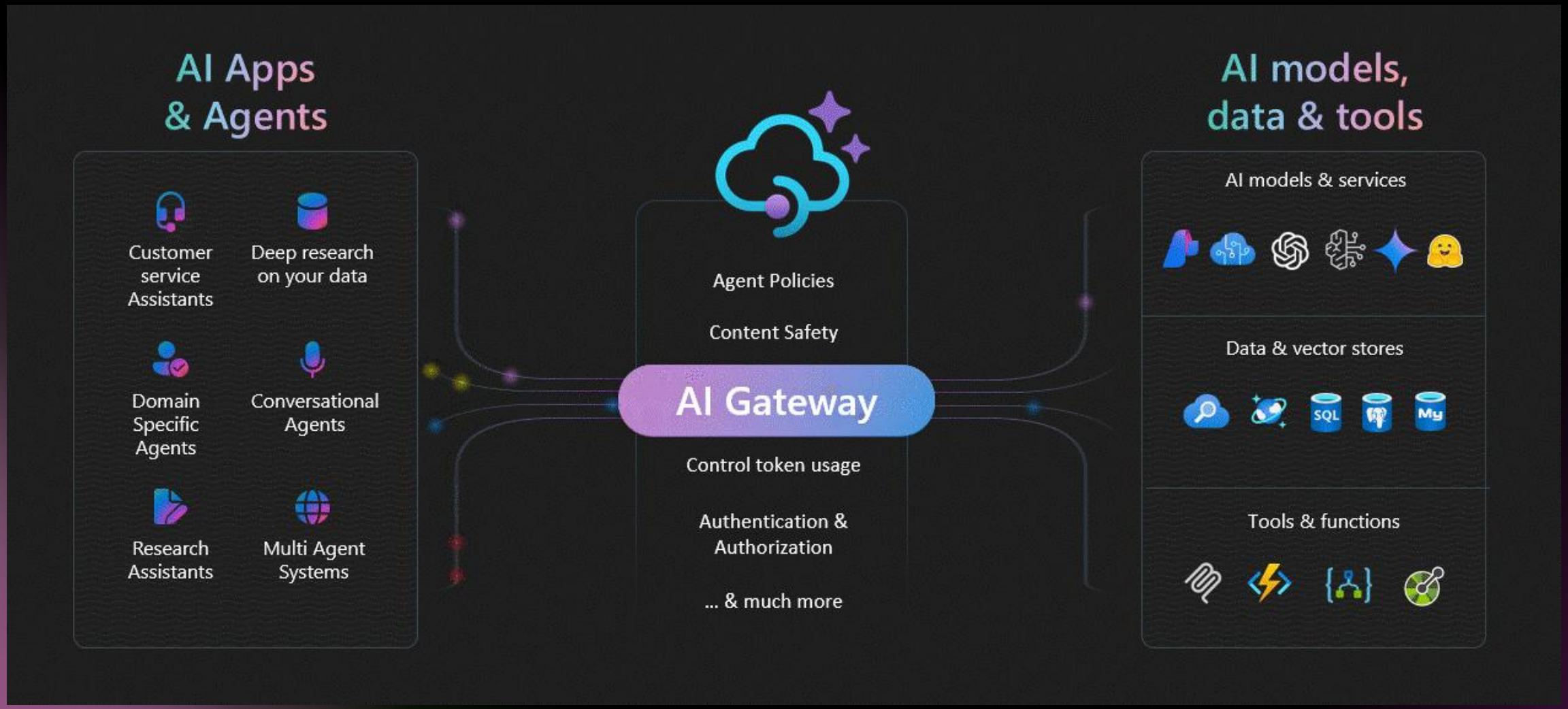


Azure AI Gateway

Enterprise Features & Capabilities

Model Context Protocol (MCP) Integration with Azure API Management

Create MCP Server in minutes using APIM


AI Apps & Agents
Customer service Assistants
Deep research on your data
Domain Specific Agents
Conversational Agents
Research Assistants
Multi Agent Systems
Agent Policies
Content Safety
AI Gateway
Control token usage
Authentication & Authorization
... & much more
AI models, data & tools
AI models & services
Data & vector stores
SQL
My
Tools & functions

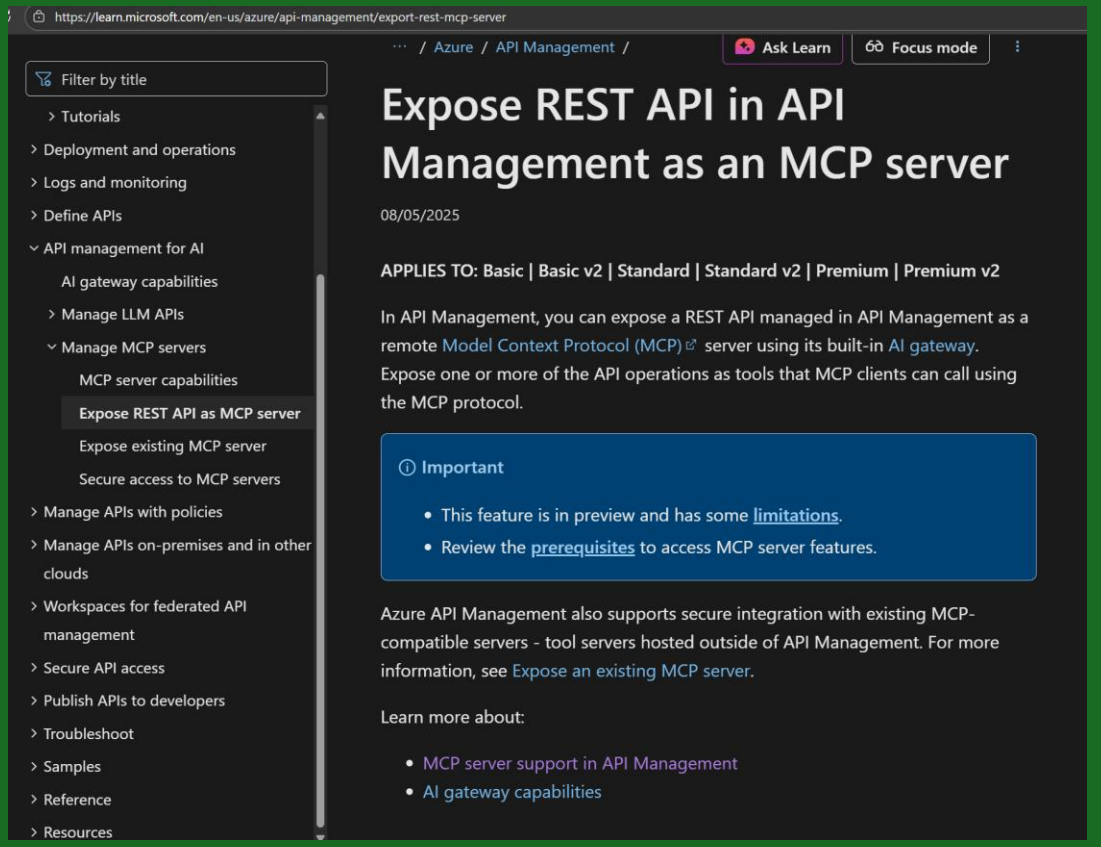
Expose API as an MCP server

Follow these steps to expose a managed REST API in API Management as an MCP server:

1. In the Azure portal, navigate to your API Management instance.
2. In the left menu, under **APIs**, select **MCP Servers** > **+ Create MCP server**.
3. Select **Expose an API as an MCP server**.
4. In **Backend MCP server**:
   a. Select a managed **API** to expose as an MCP server.
   b. Select one or more **API operations** to expose as tools. You can select all operations or only specific operations.

   > **Note**
   >
   > You can update the operations exposed as tools later in the **Tools** blade of your MCP server.

5. In **New MCP server**:
   a. Enter a **Name** for the MCP server in API Management.
   b. Optionally, enter a **Description** for the MCP server.
6. Select **Create**.



Home > apim-hello-world | MCP Servers >

Expose an API as an MCP server

Create an MCP server from an existing managed API, allowing secure use by LLMs and agents with consistent policies, access controls, and monitoring. Learn more

Backend MCP server

Choose an existing API to expose as an MCP server for AI agents and LLMs.

API * Swagger Petstore

API operations * [GET] Find pet by ID, [POST] Add a new pet to the store, [DELETE] Delet...

New MCP server

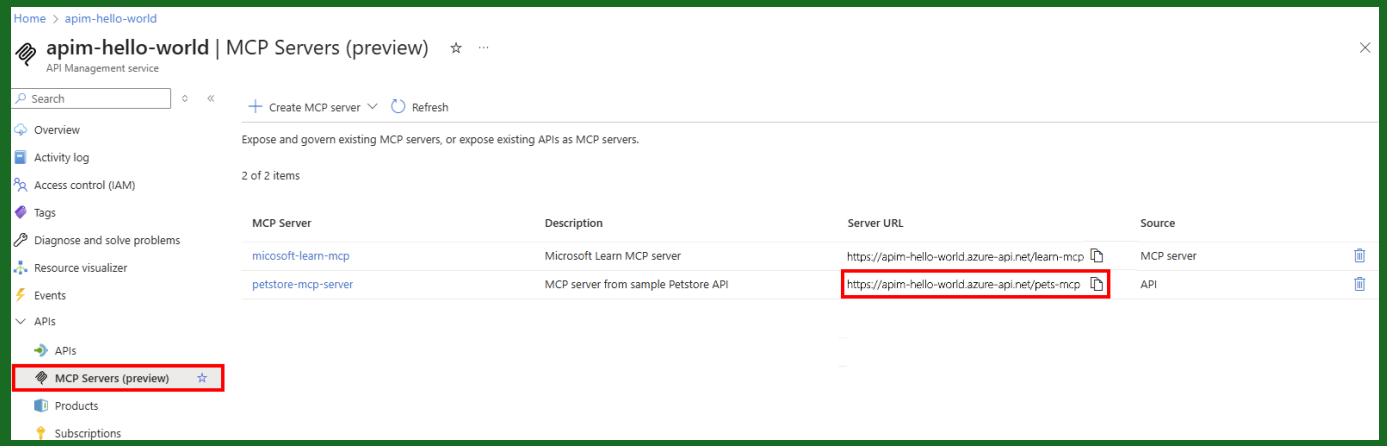
Define how the MCP server is surfaced, named, and managed through the API Management interface.

Display name * Pet server

Name * pet-server

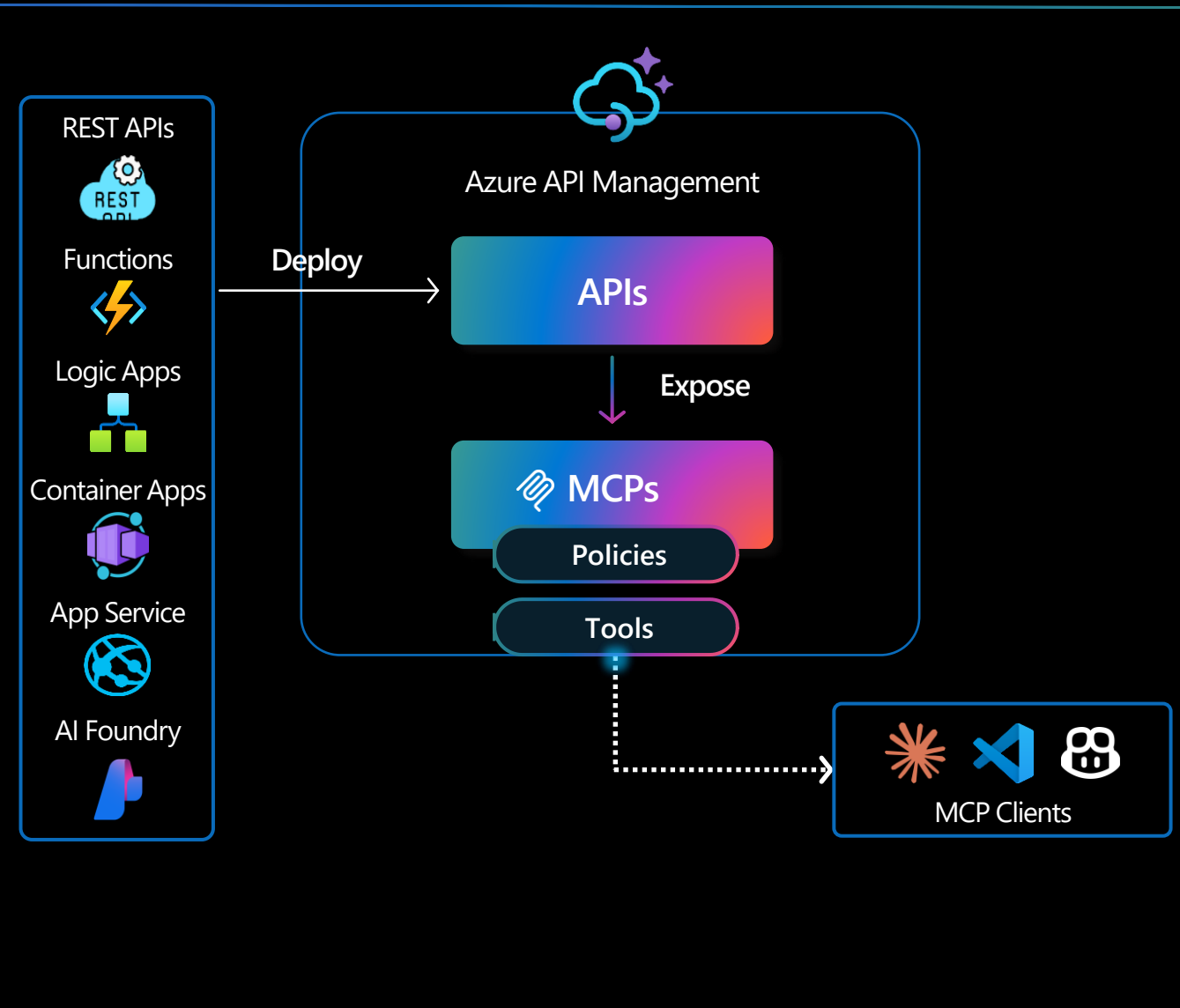
Description Sample MCP server for Petstore

Create Discard



Create MCP Servers

Scenario 1: Expose APIs as MCP servers



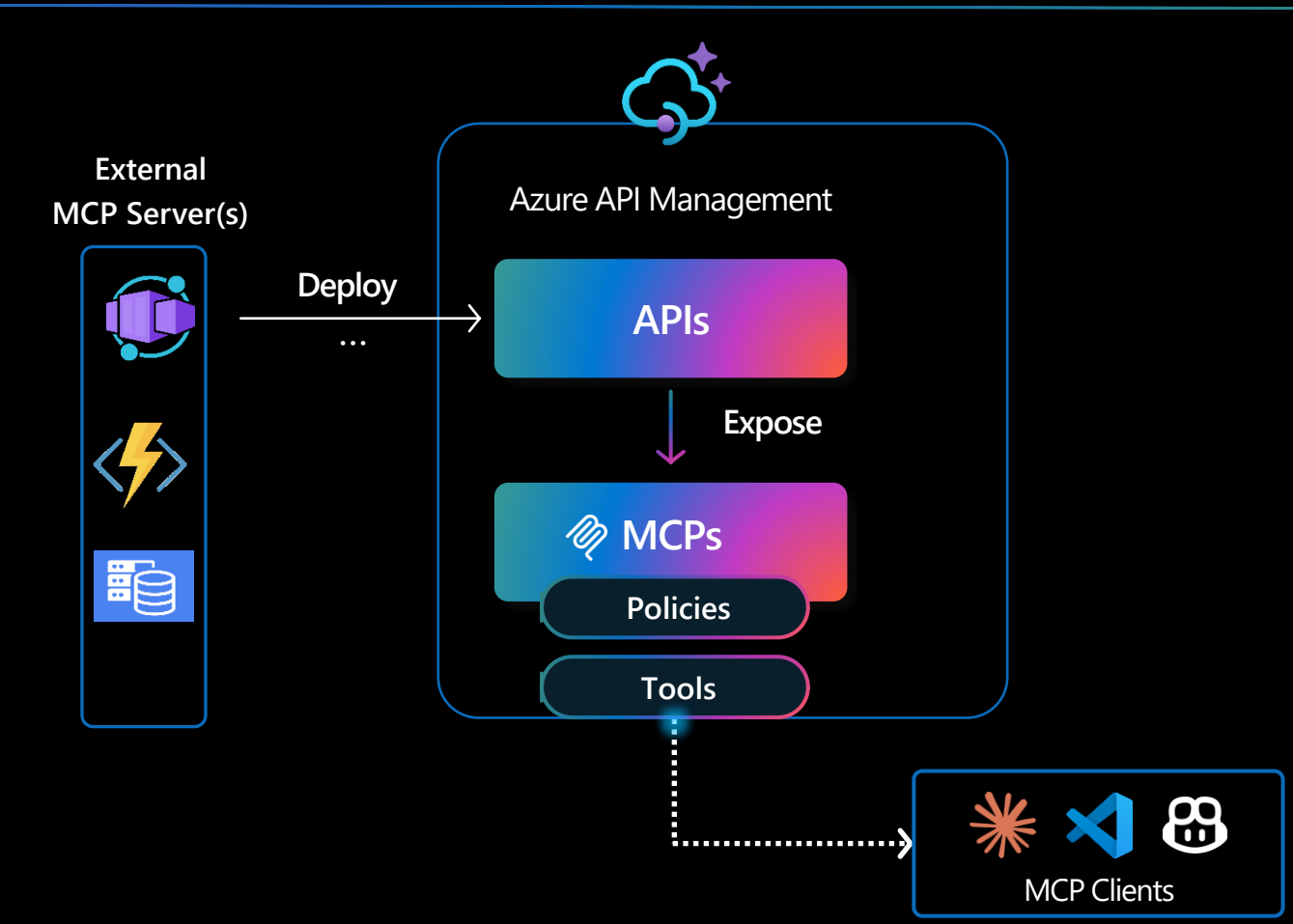
Expose any APIM-managed REST API as a remote MCP server (SSE & Streamable HTTP)

Quickly transform existing data into tools usable by Agents/ LLMs

Maintain governance, flexibility, and scalability with API Management policies

Create MCP Servers

Expose existing MCP servers

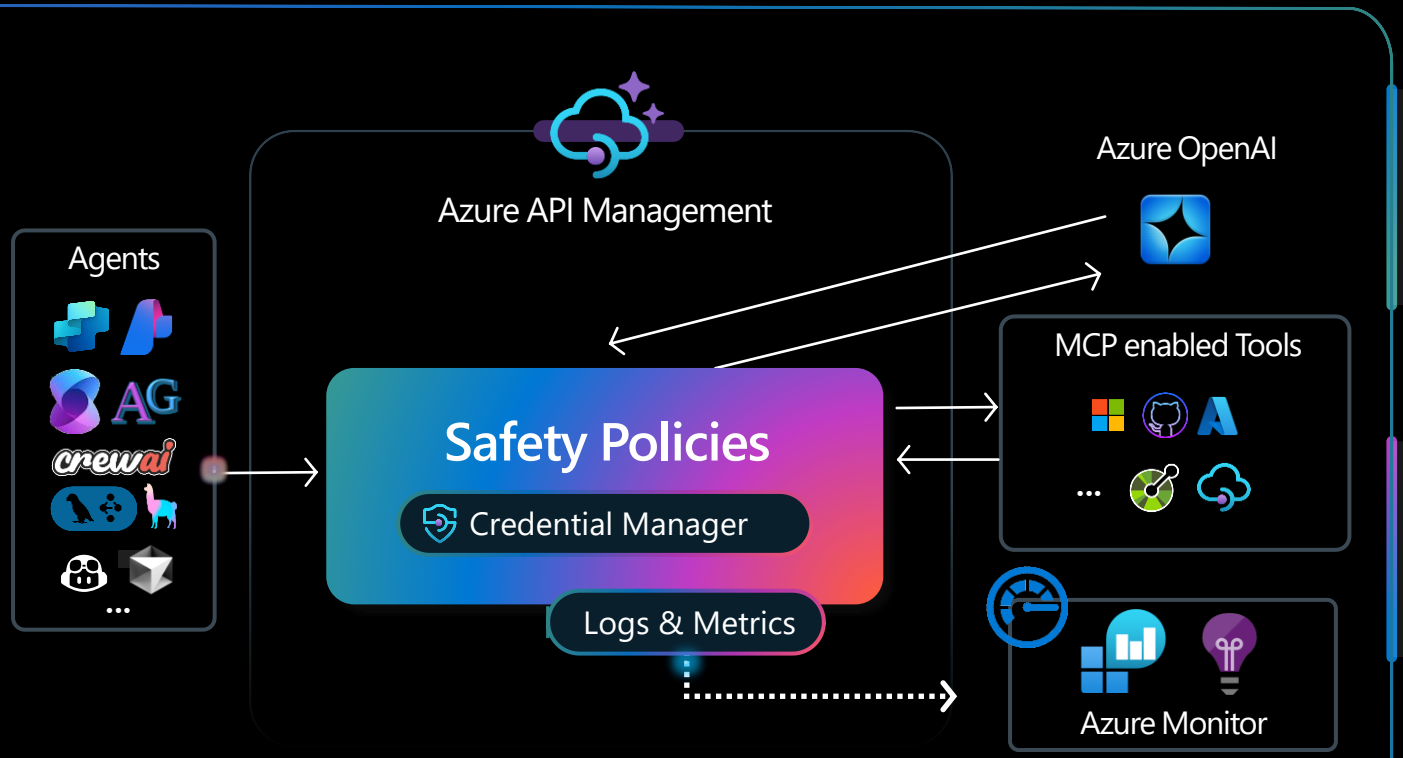


Register any external MCP-compliant server in APIM to make it available as a secure, discoverable tool for agents.

Proxy requests securely, applying OAuth 2.1, token forwarding, and Credential Manager for outbound auth.

Enforce centralized policies such as rate limits, quotas, and IP filtering—without modifying the external MCP server

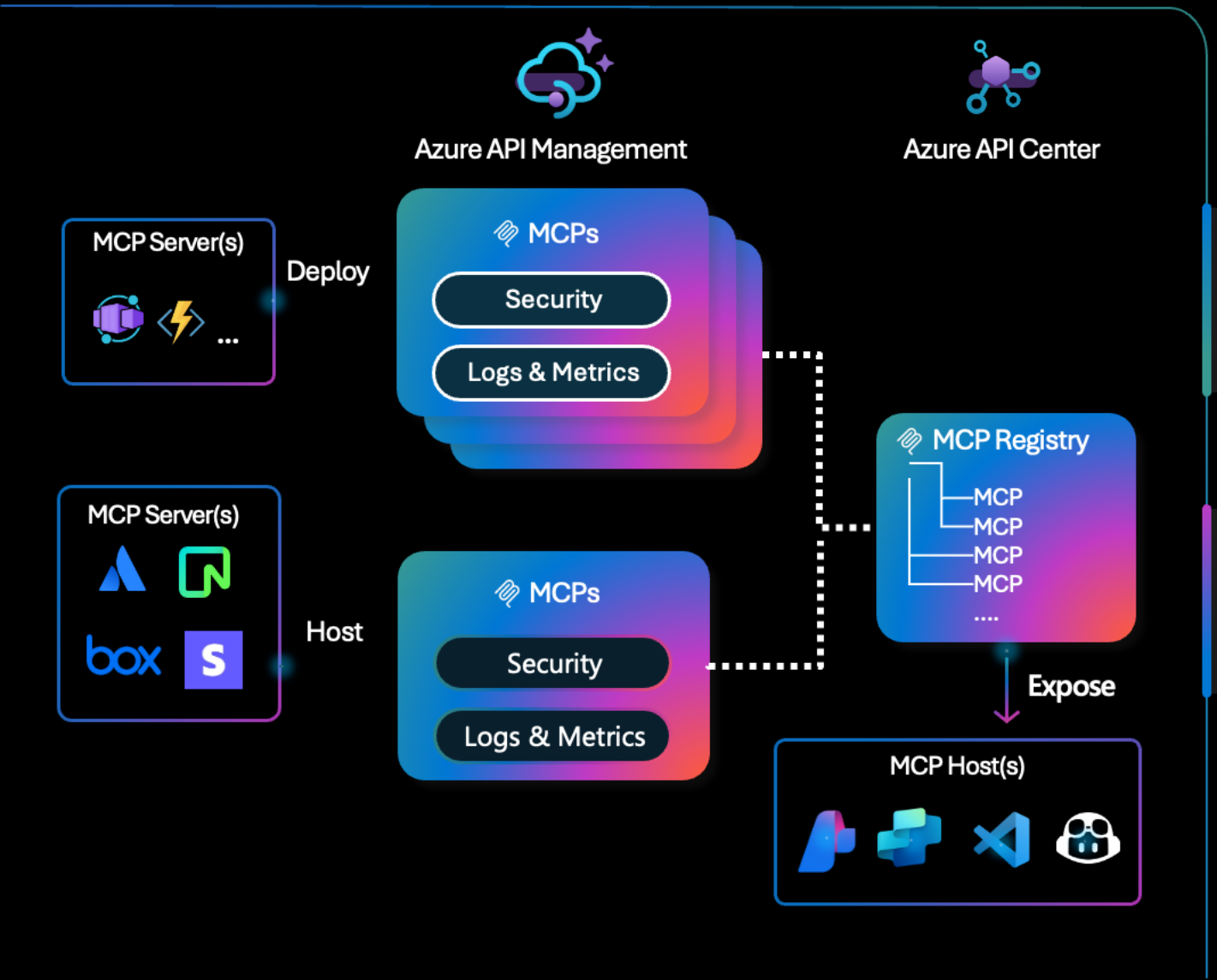
Securely Expose MCP servers using APIM



Use credential manager to authorize access to your backend MCP servers

Protect your remote MCP servers with OAuth2.1 and API Management

Discover & Govern AI Assets



Enterprises can build their own MCP Registry using Azure API Center to better discover, govern and share MCP servers within their organization

mcp.azure.com a live example demonstrating how APIC can serve as a private and enterprise-ready MCP Center

Key Enterprise Features

🔒 Security & Authentication

Policy-based access control, API key management, and OAuth support

⚡ Performance Management

Rate limiting, quota management, and traffic shaping capabilities

🛠️ Tool Management

Select and expose specific API operations as MCP tools

📊 Monitoring & Analytics

Integration with Application Insights and Azure Monitor

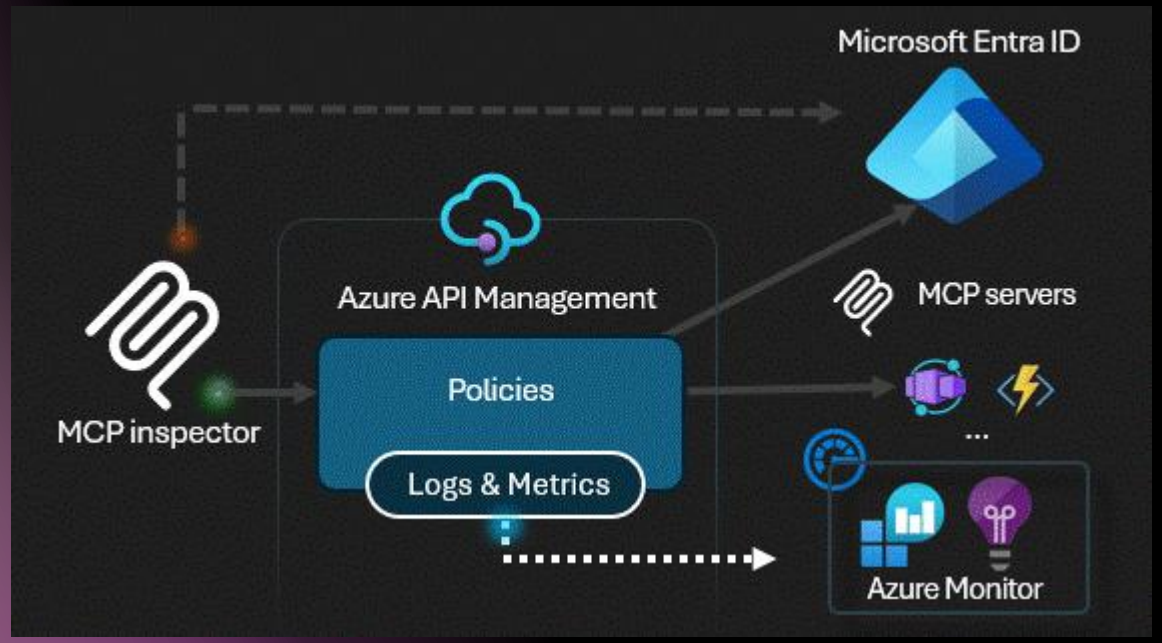
🔧 Policy Framework

XML-based policies for request/response transformation

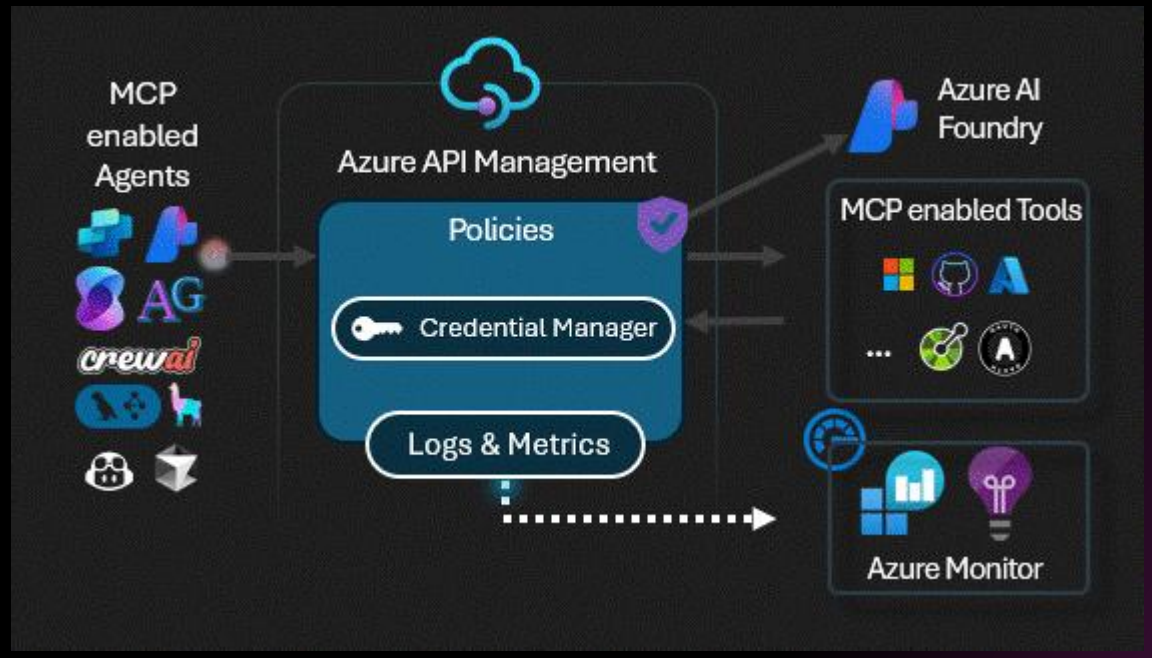
🌐 Multi-tier Support

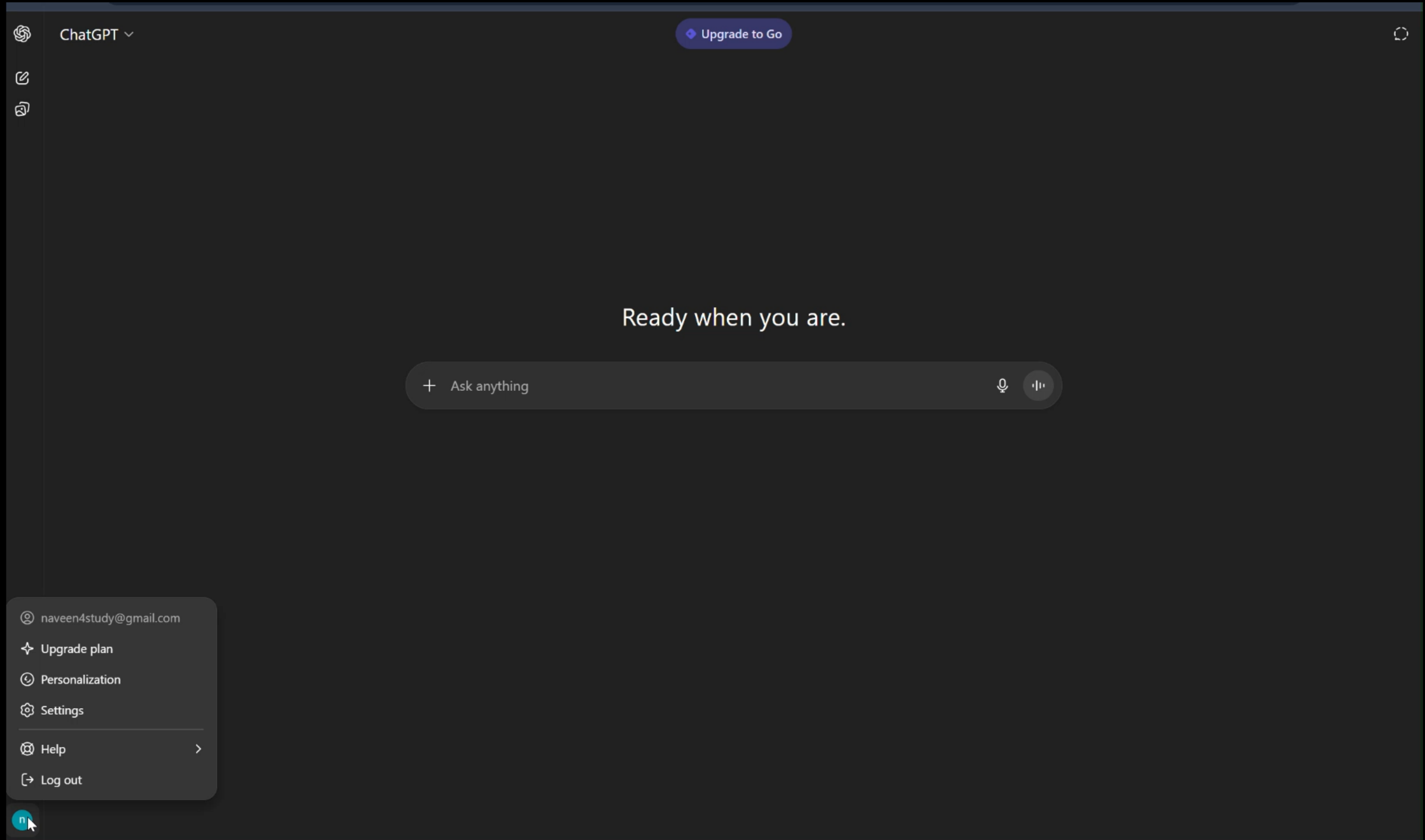
Available across Basic, Standard, and Premium tiers

MCP Client Authorization



Model Context Protocol (MCP)

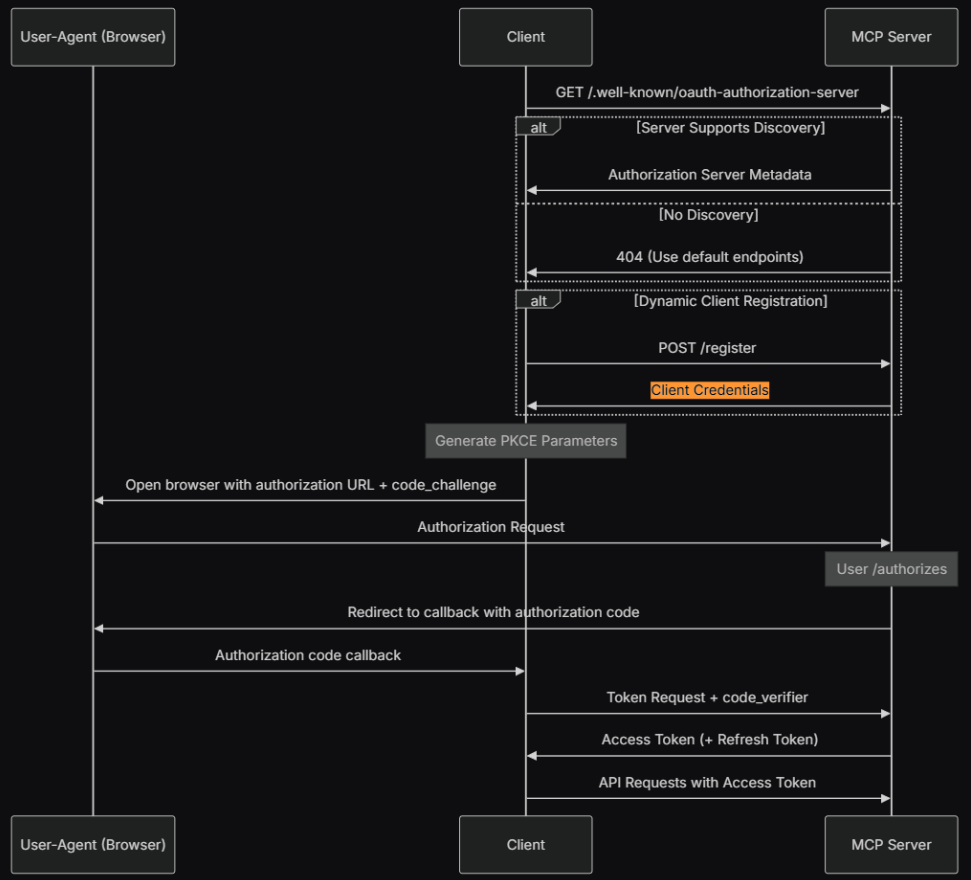


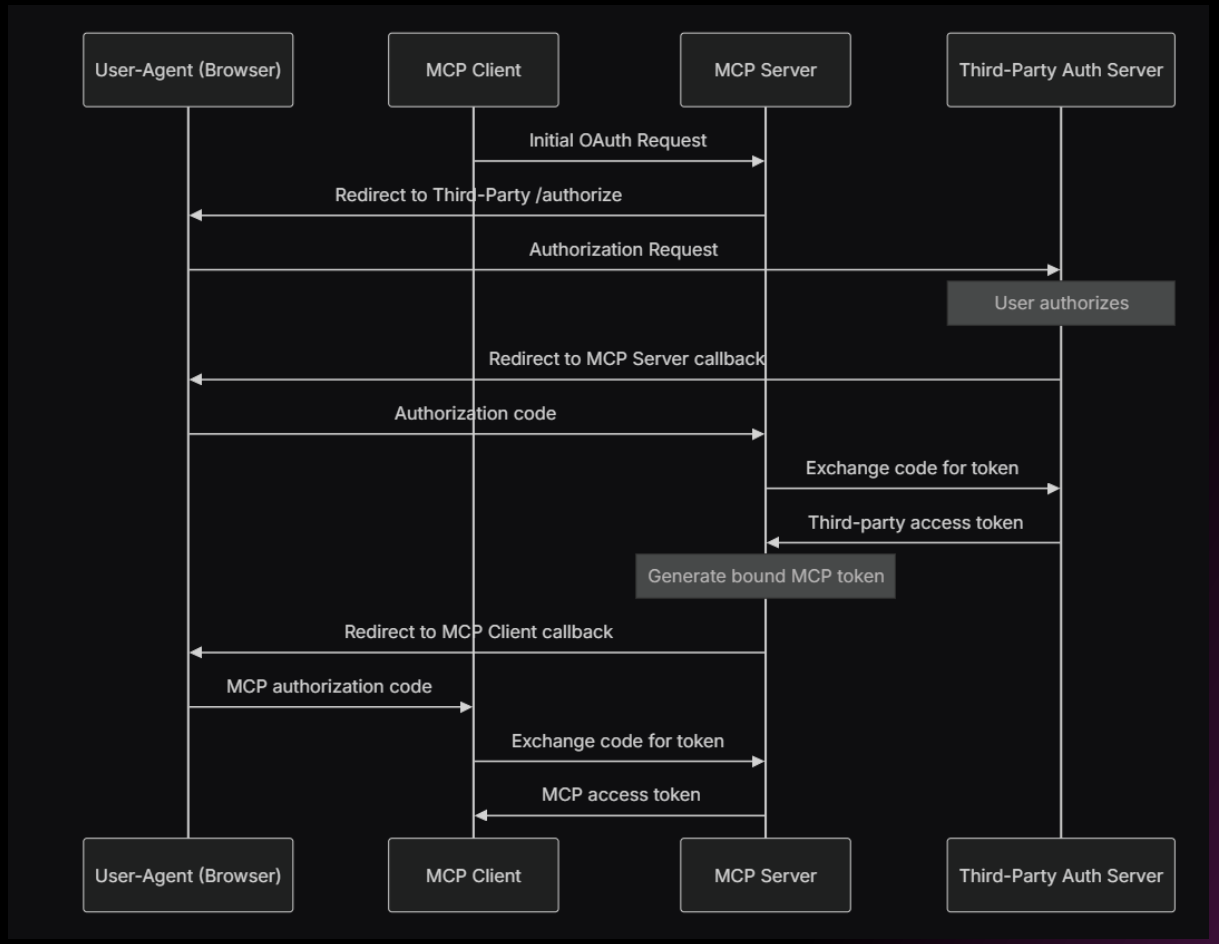

ChatGPT
Upgrade to Go
Ready when you are.
Ask anything
naveen4study@gmail.com
Upgrade plan
Personalization
Settings
Help
Log out

MCP Dynamic Client Registration

Authorization Flow Steps

The complete Authorization flow proceeds as follows:





Enterprise Security Features

Access Control & Authentication

- Subscription key authentication
- OAuth 2.0 and Azure Active Directory integration
- IP filtering and rate limiting per client
- Protected Resource Metadata (PRM) support

Policy Example: Rate Limiting

```
<rate-limit-by-key calls="5" renewal-period="30" counter-key="@(context.Request.IpAddress)" remaining-calls-variable-name="remainingCallsPerIP" />
```

Q&A Session

Frequently Asked Questions about MCP

Q Can one MCP server serve multiple clients?

A **Yes!** While each client connects to one server, a single server can handle multiple client connections simultaneously. The one-to-one relationship is from the client's perspective.

Q How does authentication work with remote servers?

A Remote servers using **HTTP transport** support standard authentication methods including OAuth tokens, bearer tokens, API keys, and custom headers. Anthropic recommends using **OAuth** for secure token-based authentication.

Q

Can servers communicate with each other?

A

No, MCP servers don't directly communicate with each other. All coordination happens through the MCP host/client. If you need cross-server functionality, the AI application orchestrates it.

Q

What happens if a server crashes?

A

Because of the **isolated one-to-one architecture**, if one server crashes, other connections remain unaffected. The client can detect the failure and the host can handle it gracefully, potentially reconnecting or notifying the user.

Q

How do I build my own MCP server?

A

Anthropic provides **SDKs in multiple languages** (Python, TypeScript, etc.) that handle all the protocol details. You primarily focus on implementing your primitives - defining tools, resources, and prompts that make sense for your use case.

Q1: Parameters & Data Formats

Do MCP servers require additional parameters or standard formats when exposing APIs, especially with APIM? Do input parameters change?

MCP servers do not require special parameters beyond standard OAuth 2.1 and JSON-RPC 2.0 requirements. When exposing APIs through API Management (APIM), the key considerations are:

- **Standard Format:** MCP uses `JSON-RPC 2.0` for all communication between clients and servers
- **Tool Definitions:** Instead of REST endpoints, MCP servers expose tools with JSON Schema descriptions that define parameters
- **Input Parameters:** Parameters don't fundamentally change, but they're wrapped in MCP's message format with metadata about tool names and capabilities
- **With APIM:** Azure API Management can act as a gateway, handling authentication and rate limiting while forwarding MCP protocol messages unchanged

Key Point: MCP wraps your existing API logic in a standardized protocol layer. Your backend API parameters remain the same, but they're exposed through MCP's tool interface rather than REST endpoints.

Learn More:

MCP Specification - modelcontextprotocol.io/specification

How does MCP differ from API usage? Is MCP for agentification and not a replacement for normal APIs? Are there standard data-passing requirements?

MCP is NOT a replacement for APIs—it's a standardized protocol layer that sits ON TOP of APIs for AI agent interactions.

- **APIs:** Direct, predictable, developer-controlled integrations using REST/GraphQL with fixed endpoints
- **MCP:** AI-friendly protocol that enables dynamic tool discovery and autonomous agent decision-making
- **Key Difference:** APIs require manual integration code; MCP allows AI to dynamically discover and use tools without pre-programming each interaction
- **MCP Wraps APIs:** An MCP server typically calls your existing REST API internally, translating between MCP protocol and your API

When to Use Each:
• Use **APIs** for: Bulk operations, deterministic workflows, financial transactions, precise control
• Use **MCP** for: AI agent autonomy, dynamic tool selection, multi-step reasoning, natural language interactions

Data Passing: MCP requires data in JSON format following JSON-RPC 2.0 specifications. Messages include request IDs, method names, and parameters as structured JSON objects.

Learn More:

MCP vs APIs Guide - tinybird.co

Is authorization handled at the API level or by MCP? What's the difference between authentication and authorization in MCP and APIM context?

Authorization is handled at the API level, NOT by MCP itself. MCP servers act as OAuth 2.1 resource servers.

- **Authentication:** Verifying WHO the user is (handled via OAuth 2.1 with authorization servers like Entra ID, Auth0, Okta)
- **Authorization:** Determining WHAT the user can access (handled by validating OAuth access tokens and checking scopes/permissions)

MCP Authorization Flow:
1. MCP client initiates OAuth flow when accessing protected resources
2. User authenticates with authorization server (not MCP server)
3. Authorization server issues access token
4. MCP client includes token in requests to MCP server
5. MCP server validates token and enforces permissions

With APIM: Azure API Management acts as an authorization gateway:

- Validates OAuth tokens (e.g., `validate-azure-ad-token` policy)
- MCP server focuses on business logic, not auth infrastructure
- Separates concerns: APIM handles auth/rate limiting, MCP handles AI tool logic

Critical Security Note: MCP servers must NEVER pass through tokens to downstream APIs. They must obtain separate tokens for upstream services to prevent confused deputy attacks.

Learn More:

Secure MCP Servers - Microsoft Learn

How is rate limiting managed for MCP servers? Is it handled at the API/APIM level or by MCP itself?

Rate limiting is handled at the API Gateway/APIM level, NOT by MCP protocol itself.

Why External Rate Limiting:
- MCP servers should remain lightweight and stateless
- API gateways provide centralized traffic management
- Prevents "Denial of Wallet" attacks (runaway AI agent costs)
- Protects both the MCP server and downstream APIs

Implementation Options:

- **Azure API Management:** Use built-in rate limiting policies (e.g., `rate-limit` or `quota` policies per subscription key, IP, or user)
- **Apache APISIX:** Configure rate limiting plugins with Redis for distributed tracking
- **Application Level:** Implement rate limiting in your MCP server code if needed for tool-specific constraints

Best Practices:

- Set different rate limits per client or subscription tier
- Use sliding window algorithms for accurate limiting
- Return HTTP 429 (Too Many Requests) with Retry-After headers
- Monitor token usage for LLM API calls to prevent cost overruns
- Implement circuit breakers for downstream service protection

Learn More:

API Gateway for MCP - API7.ai

How do MCP client-server relationships work? What is the one-to-one connection? What is the role of MCP host?

MCP follows a client-host-server architecture with specific relationships:

Three Core Components:
1. **MCP Host:** The AI application users interact with (e.g., Claude Desktop, Cursor IDE, Windsurf)
2. **MCP Client:** Lives inside the host, manages connections to servers
3. **MCP Server:** Exposes tools, resources, and prompts

Key Relationship Rules:

- **1:1 Connection:** Each MCP client maintains a dedicated connection to ONE specific MCP server
- **1:Many at Host Level:** A single host can manage MULTIPLE clients, each connecting to different servers
- **Stateful Session:** Client-server connections are persistent and maintain context throughout the session

Host Responsibilities:

- Creates and manages multiple client instances
- Maintains security boundaries between servers
- Coordinates tool selection and execution
- Passes user prompts to the LLM and interprets responses

Example: In Claude Desktop connecting to GitHub, Slack, and Google Drive:

- Claude Desktop = 1 Host
- 3 separate Clients (one for each service)
- 3 separate MCP Servers (GitHub server, Slack server, Drive server)

How do you handle multiple MCP servers? Can one client connect to multiple servers?

A single MCP client CANNOT connect to multiple servers. Each client connects to exactly ONE server.

> **Handling Multiple Servers:**
> The MCP host application creates separate client instances for each server connection. Think of it like having multiple browser tabs—each tab (client) connects to one website (server).

Architecture Pattern:

- 1 Host → N Clients → N Servers (where each client-server is 1:1)
- Host orchestrates which client to use based on the required tool
- Clients operate independently with isolated sessions
- No direct communication between servers

Configuration Example (Claude Desktop):

- Edit `claude_desktop_config.json`
- Define multiple server entries with commands to launch each
- Each server runs as a separate process
- Host automatically routes tool requests to the appropriate server

Benefits of This Design:

- **Security:** Servers are isolated; compromise of one doesn't affect others
- **Modularity:** Add/remove servers without affecting existing ones
- **Performance:** Servers can scale independently
- **Fault Tolerance:** Failure in one server doesn't crash the entire system

Learn More:

How MCP Servers Work - WorkOS

How is token usage managed? How do you pass context efficiently between MCP components?

Token management is critical for both cost control and performance optimization in MCP systems.

LLM Token Considerations:

- **Context Window:** MCP helps optimize what goes into the LLM's context (e.g., Claude 3.5 Sonnet: 200K tokens)
- **Tool Descriptions:** Each MCP tool includes JSON Schema descriptions that consume tokens when presented to the LLM
- **Minimize Tools:** Expose only essential tools to reduce token usage from documentation
- **Efficient Results:** Return concise, structured data from tools to minimize response tokens

OAuth Access Token Management:

- Use short-lived tokens (minutes to hours, not days)
- Implement refresh token rotation
- Store tokens securely, never in logs or URLs
- Apply principle of least privilege with scoped tokens

Context Passing Strategies:
- **Stateful Sessions:** MCP maintains session context between requests
- **Resources:** Use MCP resources to provide reusable context chunks
- **Prompts:** Pre-defined prompt templates reduce repetitive context
- **Selective Data:** Only fetch and return necessary information

Cost Optimization Tips:

- Choose appropriate LLM models (e.g., GPT-4o-mini has higher TPM limits and lower cost)
- Implement pagination for large datasets
- Use summarization for long documents
- Cache frequently accessed data
- Monitor token consumption per request

Learn More:

Token Management - Gravitee

What are common developer questions and best practices for building MCP servers?

Common Pitfalls to Avoid:

- ❌ **Token Passthrough:** Never pass client tokens directly to downstream APIs (security risk)
- ❌ **Too Many Tools:** Exposing dozens of tools increases token usage and confuses the LLM
- ❌ **Missing Error Handling:** Always handle and return meaningful errors to the client
- ❌ **Stateless Assumptions:** MCP sessions are stateful; design accordingly
- ❌ **No Rate Limiting:** Protect against runaway agent costs

✅ **Best Practices:**

- **Start Small:** Begin with stdio transport for local development, scale to HTTP/SSE for production
- **Use SDKs:** Leverage official Python, TypeScript, C#, or Java SDKs instead of building from scratch
- **Security First:** Implement OAuth 2.1, validate all tokens, use HTTPS, apply RBAC
- **Optimize Responses:** Return structured, concise data; avoid verbose text
- **Monitor Everything:** Track token usage, API calls, errors, and latency
- **Test Thoroughly:** Test with various LLM models; behavior can vary
- **Documentation:** Provide clear tool descriptions; the LLM uses these to make decisions

Development Workflow:
1. Define your tools with clear JSON Schema descriptions
2. Implement tool logic with proper error handling
3. Test locally using Claude Desktop or similar host
4. Add authentication and authorization
5. Deploy behind API gateway for production
6. Monitor and optimize based on real usage

Resources: Check official MCP GitHub repo for server examples, join community discussions, and review the specification regularly as it evolves.

Learn More:

MCP GitHub - Official Resources

Agent2Agent Protocol

A new era of **agent interoperability**. Enable seamless collaboration between AI agents across different platforms and frameworks.

🤝 What is A2A Protocol?

The **Agent2Agent (A2A) Protocol**, an open standard designed to enable seamless communication and collaboration between AI agents.

Originally developed by **Google** and now donated to the **Linux Foundation**, A2A provides the definitive common language for agent interoperability in a world where agents are built using diverse frameworks and by different vendors.

Google → Linux Foundation

🌐 **Cross-Platform Communication**

🔗 **Open Standard**

🚀 **Seamless Collaboration**

The Common Language for AI Agent Interoperability

Why Use the A2A Protocol?

Addressing Key Challenges in AI Agent Collaboration

🌎 The Ch

Consider a user

Flight Boo

⚠️ Problems Without A2A

Agent Exposure

Developers often wrap agents as tools, similar to how tools are exposed in MCP. However, this is inefficient because agents are designed to negotiate directly. Wrapping agents as tools limits their capabilities.

Custom Integrations

Each interaction requires custom, point-to-point solutions, creating significant engineering overhead and maintenance burden.

Slow Innovation

Bespoke development for each new integration slows innovation and prevents rapid deployment of new agent capabilities.

Scalability Issues

Systems become difficult to scale and maintain as the number of agents and interactions grows exponentially.

Interoperability

This approach limits interoperability, preventing the organic formation of complex AI ecosystems across vendors and frameworks.

Security Gaps

Ad hoc communication often lacks consistent security measures, creating vulnerabilities and compliance risks.

✨ The A2A protocol addresses these challenges by establishing interoperability for AI agents to interact reliably and securely

Why Use the A2A Protocol?

Addressing Key Challenges in AI Agent Collaboration

🌎 The Challenge: Planning an International Trip

Consider a user request for

Flight Booking Agent

⚠️ Problems Without A2A

Agent Exposure

Developers often wrap agents as tools, similar to how tools are exposed in MCP. However, this is inefficient because agents are designed to negotiate directly. Wrapping agents as tools limits their capabilities.

Custom Integrations

Each interaction requires custom, point-to-point solutions, creating significant engineering overhead and maintenance burden.

Slow Innovation

Bespoke development for each new integration slows innovation and prevents rapid deployment of new agent capabilities.

Scalability Issues

Systems become difficult to scale and maintain as the number of agents and interactions grows exponentially.

Interoperability

This a
forma
frame

Security Gaps

✨ The A2A protocol addresses these challenges by establishing interoperability for AI agents to interact reliably and securely

🤝 Understanding A2A Protocol

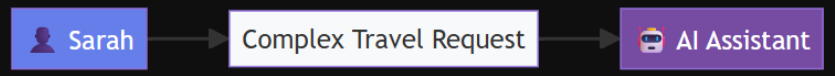
Agent-to-Agent Communication in AI Systems

📝 Step 1: User Makes a Request

Sarah wants to plan a vacation to Tokyo, Japan. She opens her AI assistant and types a complex request that requires multiple specialized services to work together.

Real-World Example

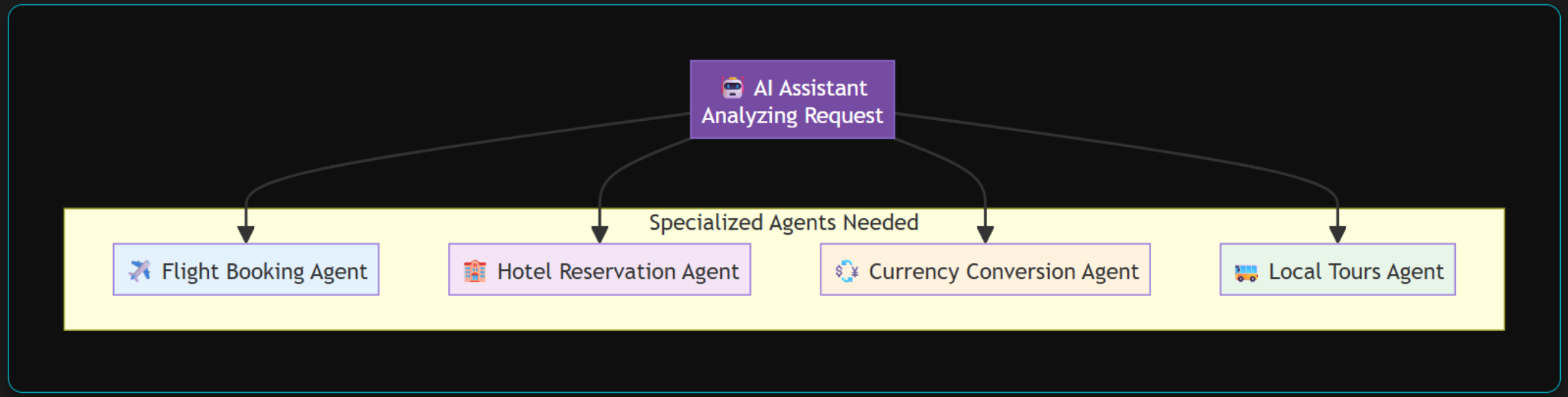
Sarah: "I want to plan a 5-day trip to Tokyo next month. Find me flights from New York, book a hotel near Shibuya, tell me how much 2000 USD converts to Japanese Yen, and recommend some popular local tours."



🔍 Step 2: AI Identifies Specialized Agents

The AI assistant analyzes Sarah's request and realizes it needs help from multiple specialized agents. Each agent has unique capabilities:

- ✈️ **Flight Booking Agent:** Searches airlines, compares prices, handles reservations
- 🏨 **Hotel Reservation Agent:** Finds accommodations, checks availability, manages bookings
- 💱 **Currency Conversion Agent:** Provides real-time exchange rates and conversion calculations
- 🚌 **Local Tours Agent:** Recommends activities, cultural experiences, and guided tours



⚠️ Step 3: The Interoperability Challenge

🚫 Without A2A Protocol - Agents Are Isolated

Each agent was built independently with different:

- Communication formats (JSON, XML, proprietary protocols)
- Authentication methods (API keys, OAuth, custom tokens)
- Data structures and naming conventions
- No way to discover what c

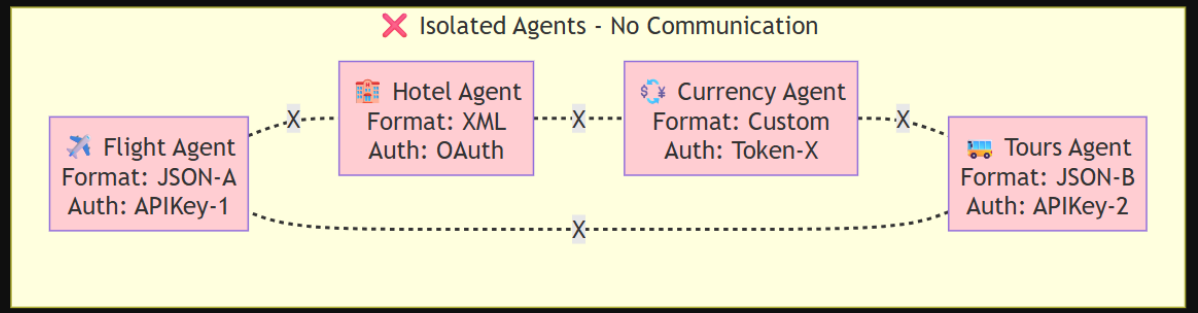
The Result: The AI assistant receives fragments of information but cannot coordinate the agents effectively. Sarah gets an incomplete response with inconsistent data, or the assistant must manually translate between each agent's unique format - a slow and error-prone process.

What Goes Wrong

```
Flight Agent to Hote
Hotel Agent: ❌ ERRO

Currency Agent to To
Tours Agent: ❌ ERRO

AI Assistant to Flig
Flight Agent: ❌ No
```

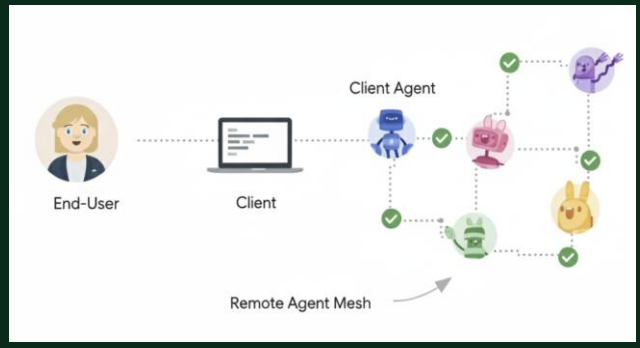


✅ Step 4: A2A Protocol - The Solution

🤝 With A2A Protocol - Seamless Collaboration

A2A provides a standardized protocol where agents can:

- **Discover** each other's capabilities automatically
- **Communicate** using a common message format
- **Share** data in standardized structures
- **Authenticate** securely using agreed methods
- **Coordinate** complex multi-agent workflows



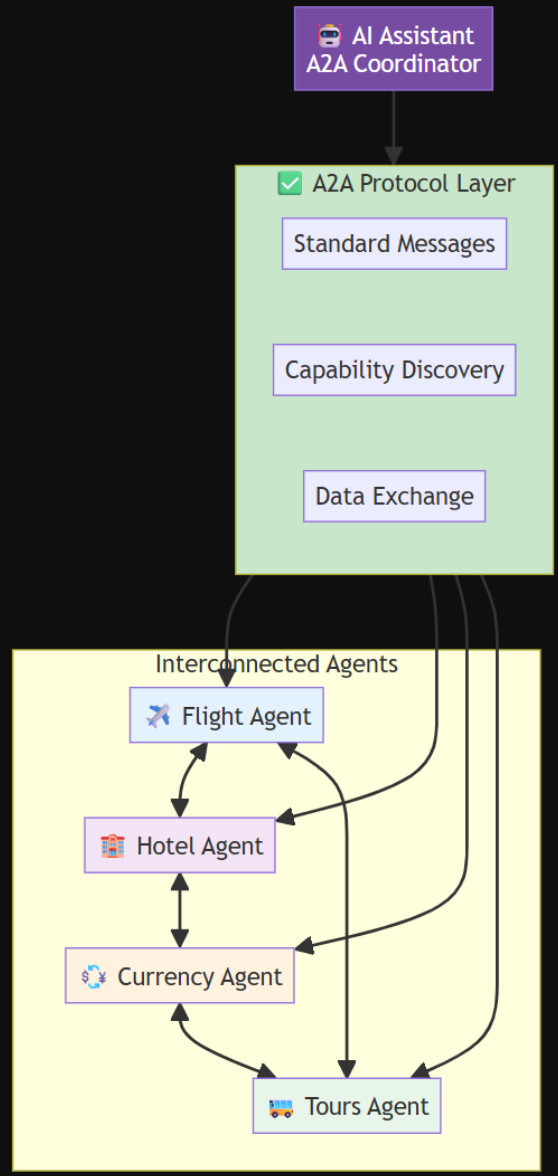
How It Works with A2A

AI Assistant: "Query all agents for Tokyo travel on specified dates"

```
✅ Flight Agent: "I can provide: searchFlights(), bookTicket(), priceCompare()"
✅ Hotel Agent: "I can provide: findHotels(), checkAvailability(), makeReservation()"
✅ Currency Agent: "I can provide: getExchangeRate(), convertCurrency()"
✅ Tours Agent: "I can provide: listTours(), getReviews(), bookActivity()"
```

```
Flight Agent → Hotel Agent: "User arrives 6:00 PM, needs check-in after"
Hotel Agent → Tours Agent: "Hotel in Shibuya, recommend nearby activities"
Currency Agent → All: "Current rate: 1 USD = 149.23 JPY, 2000 USD = 298,460 JPY"
```

AI Assistant to Sarah: "I've coordinated everything for your Tokyo trip! Found round-trip flights ($850), a hotel in Shibuya ($120/night), and 3 popular tours. Your 2000 USD converts to 298,460 JPY. Would you like to proceed with booking?"



A2A Solution Benefits

How A2A Tran

Secure Collaboration

Uses HTTPS for secure communication and maintains opaque operations, so agents can't see the inner workings of other agents.

Agent Autonomy

Allows agents to retain their individual capabilities and act as autonomous entities while collaborating with other agents.

Long-Running Operations

Supports LRO and streaming with Server-Sent Events (SSE) and asynchronous execution for complex tasks.

Key Design Principles of A2A

Built for Enterprise-Grade Agent Communication

Simplicity

Leverages existing standards like HTTP, JSON-RPC, and Server-Sent Events (SSE). Avoids reinventing core technologies and accelerates developer adoption.

Enterprise Readiness

Addresses critical enterprise needs. Aligns with standard web practices for robust authentication, authorization, security, privacy, tracing, and monitoring.

Asynchronous Support

Natively supports long-running tasks. Handles scenarios where agents or users might not remain continuously connected using streaming and push notifications.

Modality Independent

Allows agents to communicate using a wide variety of content types. Enables rich and flexible interactions beyond plain text.

Opaque Execution

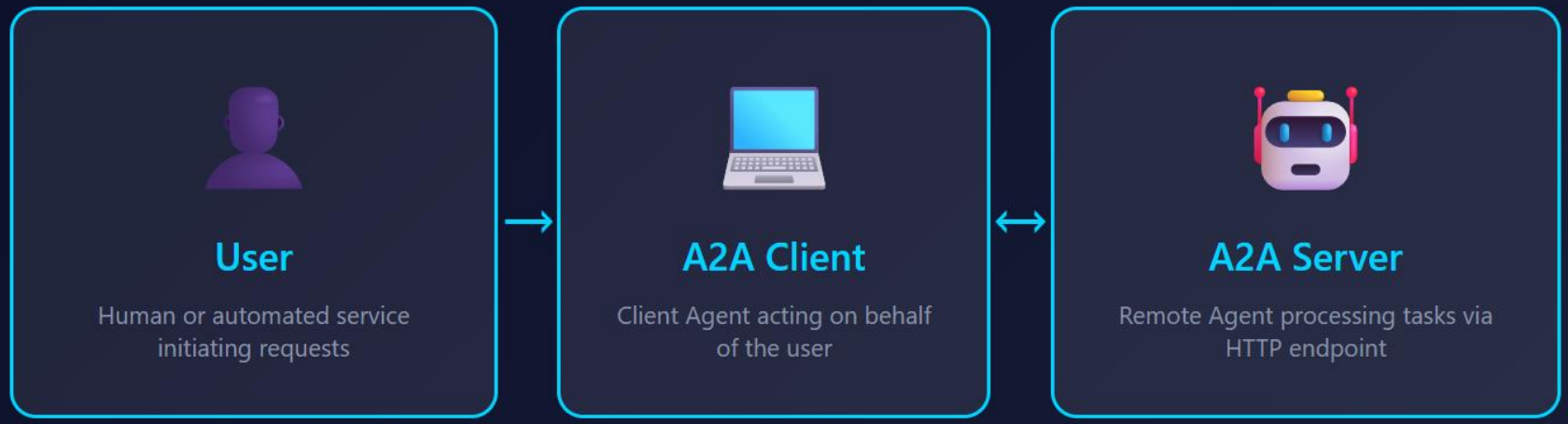
Agents collaborate effectively without exposing their internal logic, memory, or proprietary tools. Preserves intellectual property and enhances security.

Future-Proof Design

Built on principles that prioritize broad adoption, scalability, and extensibility for evolving AI agent ecosystems.

Core Actors

The Players in A2A Communication



The **A2A Server (Remote Agent)** operates as an opaque (black-box) system from the client's perspective, meaning its internal workings, memory, or tools are not exposed.

Fundamental Communication Elements

Building Blocks of A2A Interactions

Agent Card

JSON metadata describing agent's identity, capabilities, endpoint, skills, and authentication requirements. Enables discovery and secure interaction.

Task

Stateful unit of work with unique ID and defined lifecycle. Facilitates tracking of long-running operations and multi-turn interactions.

Message

Single turn of communication containing content and role ("user" or "agent"). Conveys instructions, context, questions, or status updates.

Part

Fundamental content container (TextPart, FilePart, DataPart) used within Messages and Artifacts. Provides modality flexibility.

Artifact

Tangible output generated by an agent during a task (document, image, structured data). Delivers concrete, structured, and retrievable results.

Interaction Mechanisms

Flexible Communication Patterns

Request/Response

Client sends request and server responds. For long-running tasks, client periodically polls for updates.

Streaming (SSE)

Client receives real-time, incremental results over an open HTTP connection using Server-Sent Events.

Push Notifications

Server actively sends asynchronous notifications to client webhook for very long-running or disconnected scenarios.

Key Features

- Ensures efficient and reliable information exchange
- Accommodates various task complexities and durations
- Provides flexibility in responsiveness and persistence

A2A and MCP

Complementary Protocols in the Agent Stack

Understanding the Relationship

A2A and MCP address distinct yet related aspects of agent interaction, working together in the broader AI ecosystem.

A2A Focus

Enabling agents to collaborate within their native modalities, allowing them to communicate as agents rather than being constrained to tool-like interactions. Facilitates complex, multi-turn interactions like negotiation or clarification.

MCP Focus

Reducing the complexity involved in connecting agents with tools and data. Tools are typically stateless and perform specific, predefined functions (e.g., calculator, database query).

The Agent Stack

A2A: Standardizes communication among agents deployed in different organizations and developed using diverse frameworks.

MCP: Connects models to data and external resources.

Frameworks (like ADK): Provide toolkits for constructing agents.

Models: Fundamental to an agent's reasoning, these can be any Large Language Model (LLM).

Why Agents Are Not Tools

The Critical Distinction Between A2A and MCP

The Fundamental Difference

The practice of encapsulating an agent as a simple tool is fundamentally limiting, as it fails to capture the agent's full capabilities.

Agents (A2A)

Stateful: Maintain context and memory across interactions

Autonomous: Can reason, plan, and make decisions

Multi-turn: Engage in complex negotiations and clarifications

Native modality: Communicate as agents, not as wrapped tools

Tools (MCP)

Stateless: Perform specific, predefined functions

Deterministic: Execute fixed operations without reasoning

Single-turn: Request-response pattern

Constrained: Limited to tool-like interactions

Working Together

A2A is positioned to complement MCP, not replace it. Together, they enable a complete agent ecosystem where agents can collaborate with each other (A2A) while accessing tools and data (MCP).

Frequently Asked Questions

Common Questions About A2A and MCP

Q: When should I use A2A vs MCP?

Use A2A when you need agents to collaborate autonomously with multi-turn interactions, reasoning, and negotiation. Use MCP when you need to connect agents to stateless tools and data sources with simple request-response patterns.

Q: Can I use both A2A and MCP together?

Absolutely! A2A and MCP are complementary protocols. An agent can use MCP to access tools and data while using A2A to collaborate with other agents. This creates a complete ecosystem for agent interaction.

Q: Does A2A work with any agent framework?

Yes! A2A is framework-agnostic and works with agents built using ADK, LangGraph, Crew AI, or any other framework. The protocol standardizes communication regardless of how the agent was constructed.

More Questions

Technical and Security Considerations

Q: How does A2A ensure security between agents?

A2A uses HTTPS for secure communication and implements opaque execution, meaning agents can't see each other's internal logic. It follows enterprise-grade security practices including authentication, authorization, and standard web security protocols.

Q: What happens if an agent becomes unavailable during a long-running operation?

A2A natively supports asynchronous execution and long-running operations (LRO). It uses streaming with Server-Sent Events (SSE) and push notifications to handle scenarios where agents or users might not remain continuously connected.

Q: Is A2A production-ready for enterprise use?

Yes! A2A was designed with enterprise readiness in mind from the start. It aligns with standard web practices for authentication, authorization, security, privacy, tracing, and monitoring. Originally developed by Google and now under the Linux Foundation.

LinkedIn Connect

Thank you

Github resources