

MCP for DevOps: Unleashing the Power of AI and MCP in Dev/Net/SecOps

cisco Live !

Shannon McFarland
VP, Cisco DevNet & Cisco OSPO
CCIE #5245

Agenda

- 01 What is MCP?
- 02 MCP Architecture & Core Components
- 03 MCP - Communications
- 04 MCP - Discovery
- 05 Use Cases
- 06 Summary



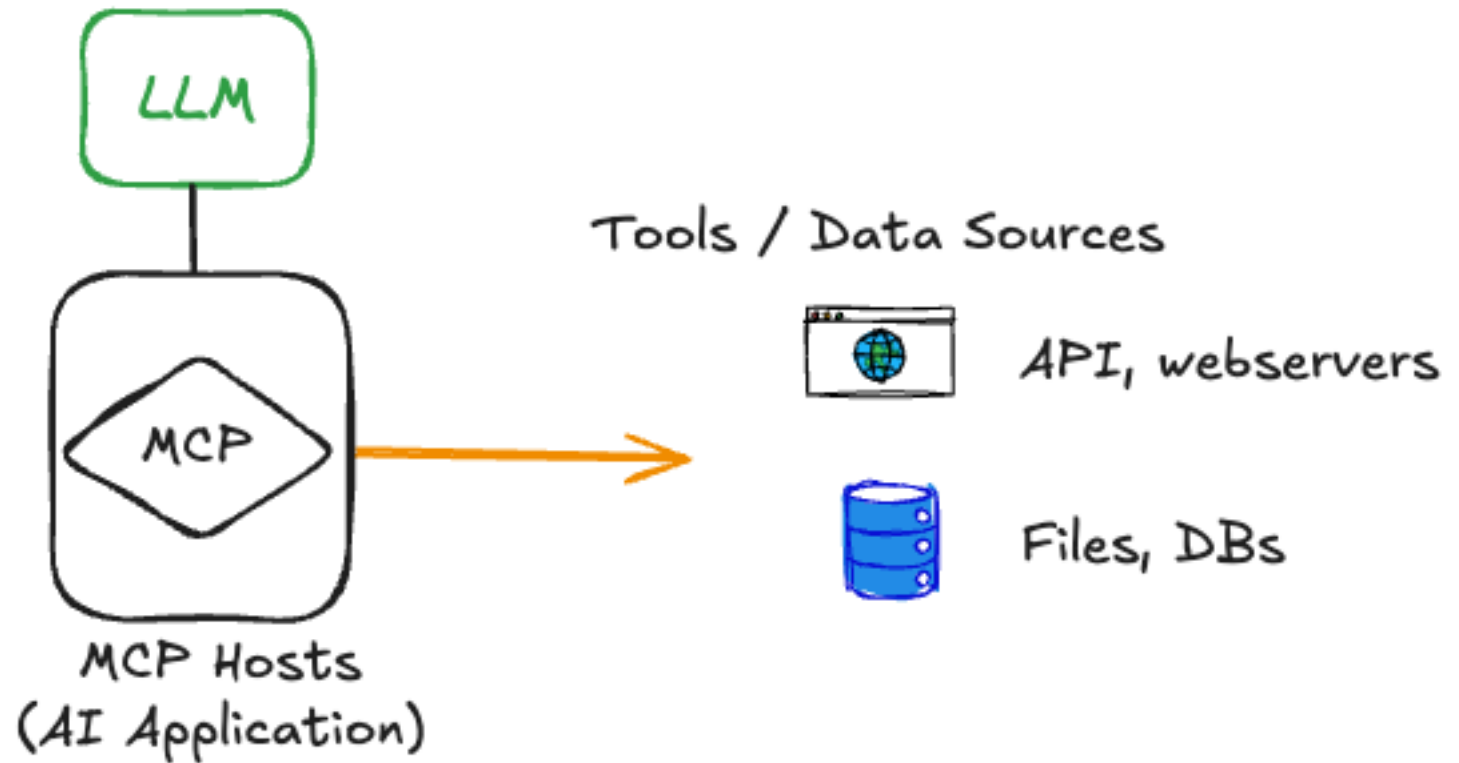
What is MCP?

What does AI think MCP is?

- Multi-cloud control protocol
- Multicast control plane
- Multiple computer programs
- Multi-cloud control plane
- many many more...

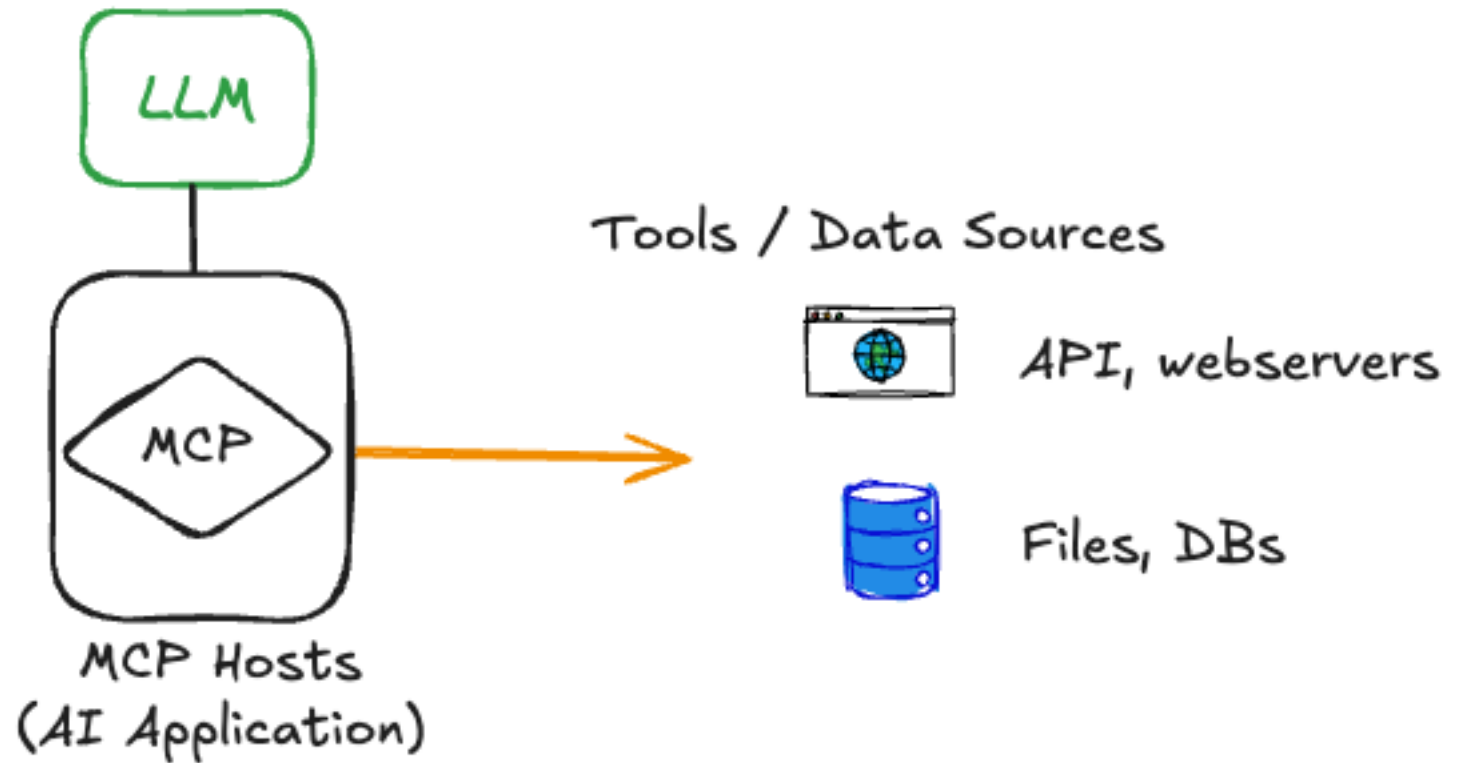
What is MCP, and Why Does it Matter?

- Model Context Protocol (MCP) is:
 - A protocol that provides a uniform way to plug in an AI model into tools and services
 - A lightweight communication protocol designed for AI agents and applications
 - Built to connect AI agents and applications to tools, APIs, databases, and filesystems
 - Structured as a client/server architecture
 - Plumbing



What is MCP, and Why Does it Matter?

- MCP is not:
 - A messaging protocol for agent-to-agent communication
 - An LLM, database, AI assistant or agent
 - A general-purpose integration platform
 - A replacement for your existing APIs or data bus



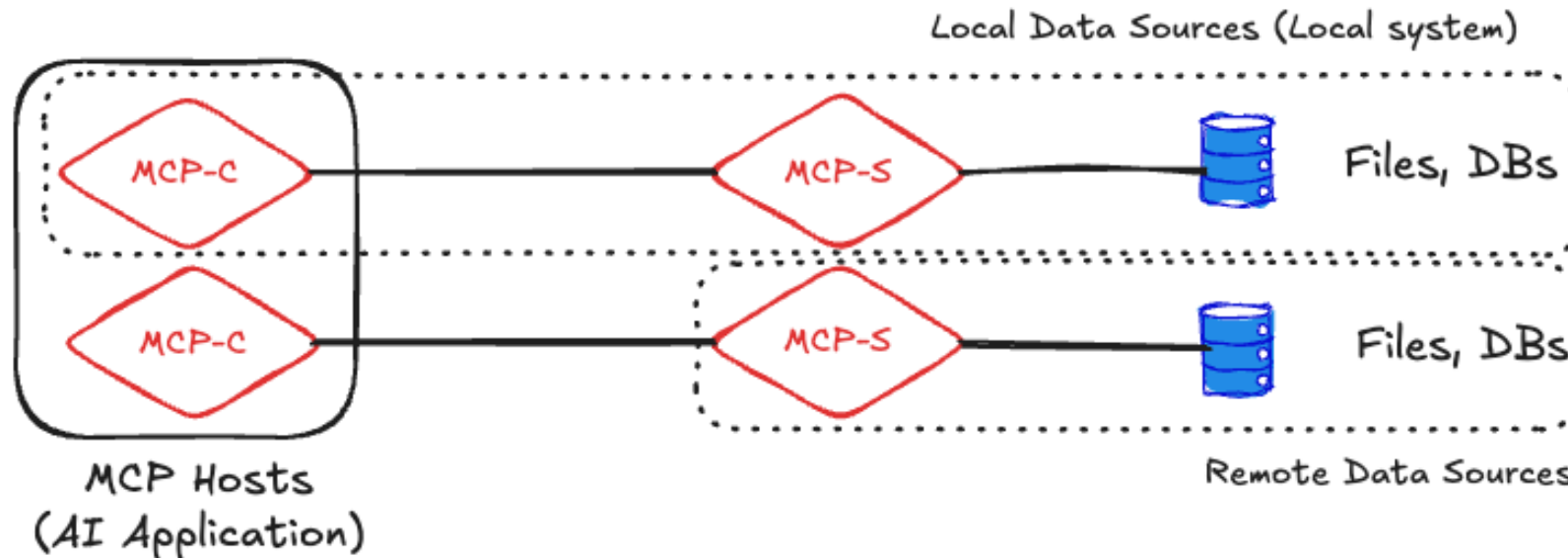
What is MCP, and Why Does it Matter?

- MCP's job is tightly scoped: give an **AI agent** a clean, **standardized** way to **discover, request, and invoke capabilities on existing tool-based infrastructure**. If your LLM-powered bot needs to call a REST API, list files, or query a database—MCP provides the glue
- **MCP matters** because it **reduces** and, in many cases, **removes the toil for AI applications and agents to find, connect to, and leverage external tools and services such as APIs, data sources, and other non-AI native tool sets**
- For **Dev/Net/SecOps** staff, it can bring immediate **value** for you to leverage an **AI agent to connect to your existing data sources and APIs** so that an **operationally-focused agent** can more accurately **complete tasks**

MCP – Architecture and Components

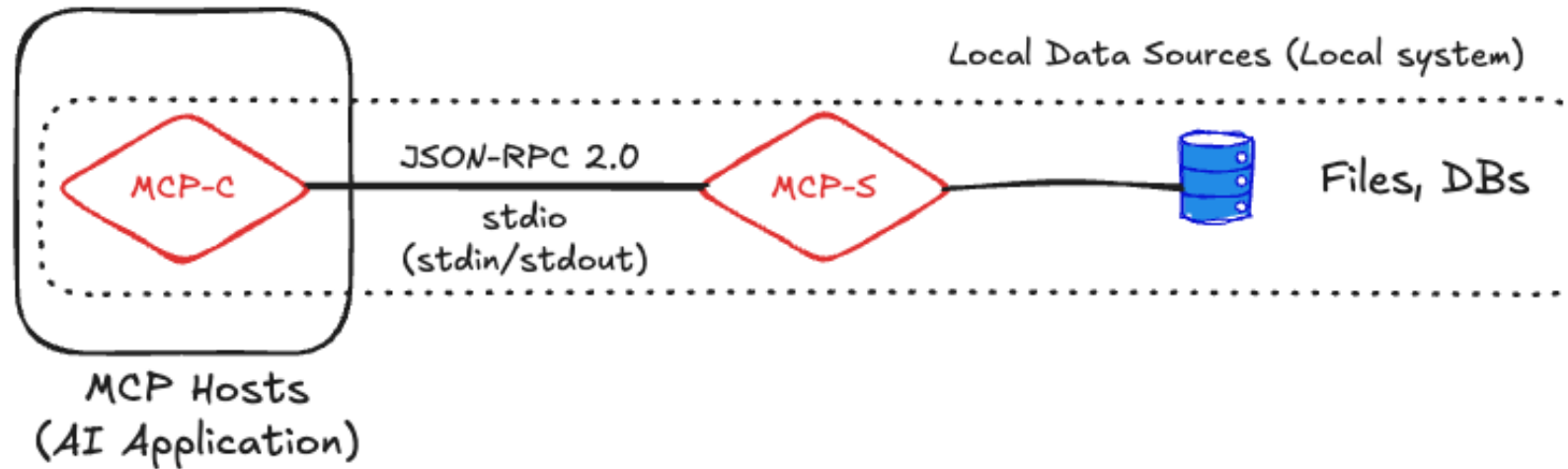
MCP – Architecture and Core Components

- **MCP Client and Server:** MCP-C software runs on MCP hosts, establishing one-to-one connections with MCP-S servers, which represent specific service or tool capabilities
- **MCP SDK Usage:** MCP hosts utilize language-specific SDKs, such as the MCP Python SDK (<https://github.com/modelcontextprotocol/python-sdk>) for client-server connections, applicable to both client-side and server-side code development
- **MCP Clients and Applications:** Many MCP clients are full applications or AI agents with native MCP client SDK functionality, with examples available at <https://modelcontextprotocol.io/clients>
- **MCP Server Listings:** Various sources provide MCP server lists, including the MCP project at <https://modelcontextprotocol.io/examples> and specific lists from providers like Cursor at <https://cursor.directory/>
- **Connection Example:** Each MCP-C instance connects one-to-one with MCP-S instances, such as an AI agent connecting to locally and remotely hosted servers for file system access



MCP – Communication

- MCP clients exchange messages with MCP servers using JSON-RPC 2.0 (as the wire format)
- For local data sources, MCP uses JSON-RPC over stdio (Standard Input/Output) as the transport
- The diagram illustrates how an MCP-C connects to a local MCP-S for file or DB access using stdio
- The MCP-S sends JSON-RPC messages to its standard output / stdout and reads from the standard input / stdin



MCP Example: MCP Inspector + Filesystem Server

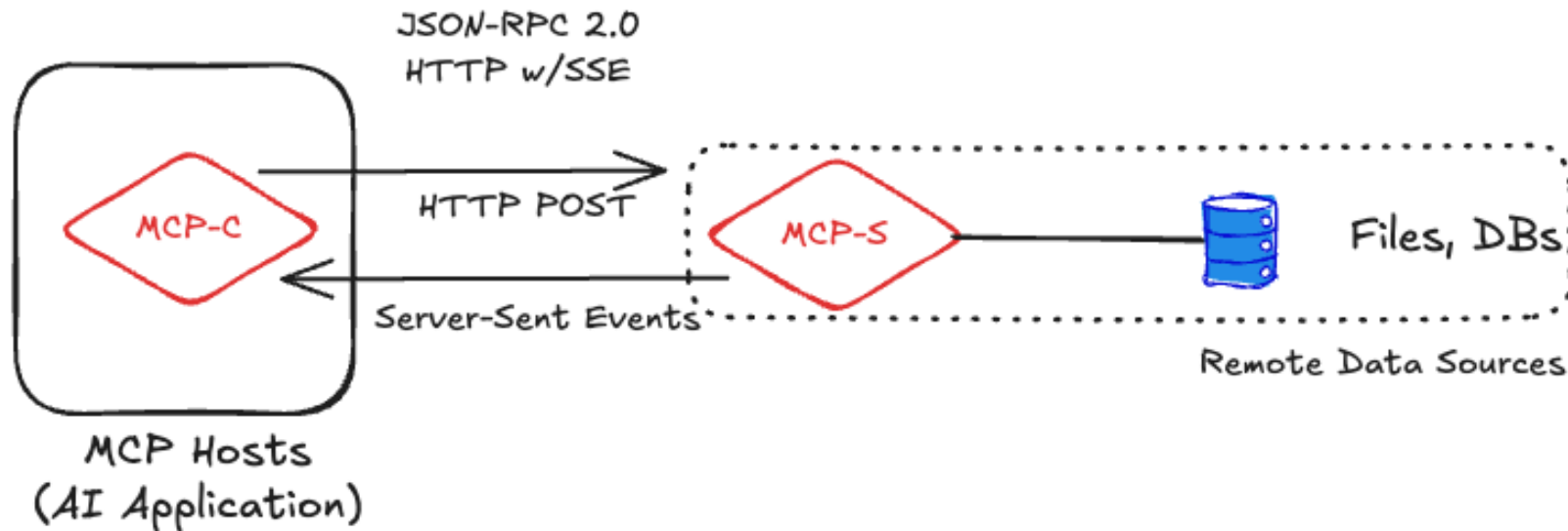
- Use the MCP Inspector tool to test the pairing of a local client (MCP inspector) with the local stdio or HTTP+SSE server
 - Example filesystem MCP server: <https://github.com/modelcontextprotocol/inspector>

```
# npx -y @modelcontextprotocol/inspector npx -y @modelcontextprotocol/server-filesystem /Users/shmcfarl/code/mcp-testing
Starting MCP inspector...
Proxy server listening on port 3000

🔍 MCP Inspector is up and running at http://localhost:5173 🚀
Query parameters: {
  transportType: 'stdio',
  command: 'npx',
  args: '-y @modelcontextprotocol/server-filesystem -y /Users/shmcfarl/code/mcp-testing',
  . . . [Output removed for clarity]
Spawned stdio transport
Connected MCP client to backing server transport
Created web app transport
Set up MCP proxy
Received message for sessionId 697bd02d-5d67-4dfc-85b9-6a12d6a99f45
Received message for sessionId 697bd02d-5d67-4dfc-85b9-6a12d6a99f45
Received message for sessionId 697bd02d-5d67-4dfc-85b9-6a12d6a99f45
```

HTTP+SSE (Server-Sent Events)

- MCP supports HTTP+SSE (Server-Sent Events) to send structured requests from service backends using MCP servers to MCP clients for local or remote connections
- The 2025-03-26 specification changes states that MCP is moving to a more flexible Streamable HTTP transport
<https://modelcontextprotocol.io/specification/2025-03-26/changelog>
- However, HTTP+SSE transport can still be used for backward compatibility - This keeps it transparent, traceable, and tool-agnostic
- **Note: As of the time of writing this blog, the new Streaming HTTP support is not completed in each SDK**
- I recommended that you read about the proposed Streamable HTTP implementation here: <https://modelcontextprotocol.io/specification/2025-03-26/basic/transport#streamable-http>
- The diagram illustrates the connection flow for HTTP+SSE scenarios. In the figure, HTTP POST is used for MCP-C -to- MCP-S messages. HTTP+SSE is used for MCP-S -to- MCP-C messages



MCP Example: MCP Message Exchange

- You can go through the MCP quick start server and client guides to learn how to setup your own weather client/server combo: <https://modelcontextprotocol.io/quickstart/server>
- Using a similar setup the weather app example, the output shows HTTP messages for stuff like a tools list call and tools call:

Tools List:

```
POST /messages/?session_id=6ccde3779adf43cc9d3f5f661508310b HTTP/1.1
Host: 127.0.0.1:8080
Accept: */*
Accept-Encoding: gzip, deflate
Connection: keep-alive
User-Agent: python-httpx/0.28.1
Content-Length: 46
Content-Type: application/json

{"method":"tools/list","jsonrpc":"2.0","id":2}
HTTP/1.1 202 Accepted
date: Tue, 08 Apr 2025 20:14:51 GMT
server: uvicorn
content-length: 8

Accepted
```

#Note: Avoid binding servers to all network interfaces (0.0.0.0):
<https://modelcontextprotocol.io/docs/concepts/transport>

Tools call:

```
. . . [Output removed for clarity]

{"method":"tools/call","params":{"name":"get_forecast","arguments":{"latitude":39.7392,"longitude":-104.9903}},"jsonrpc":"2.0","id":3}
```

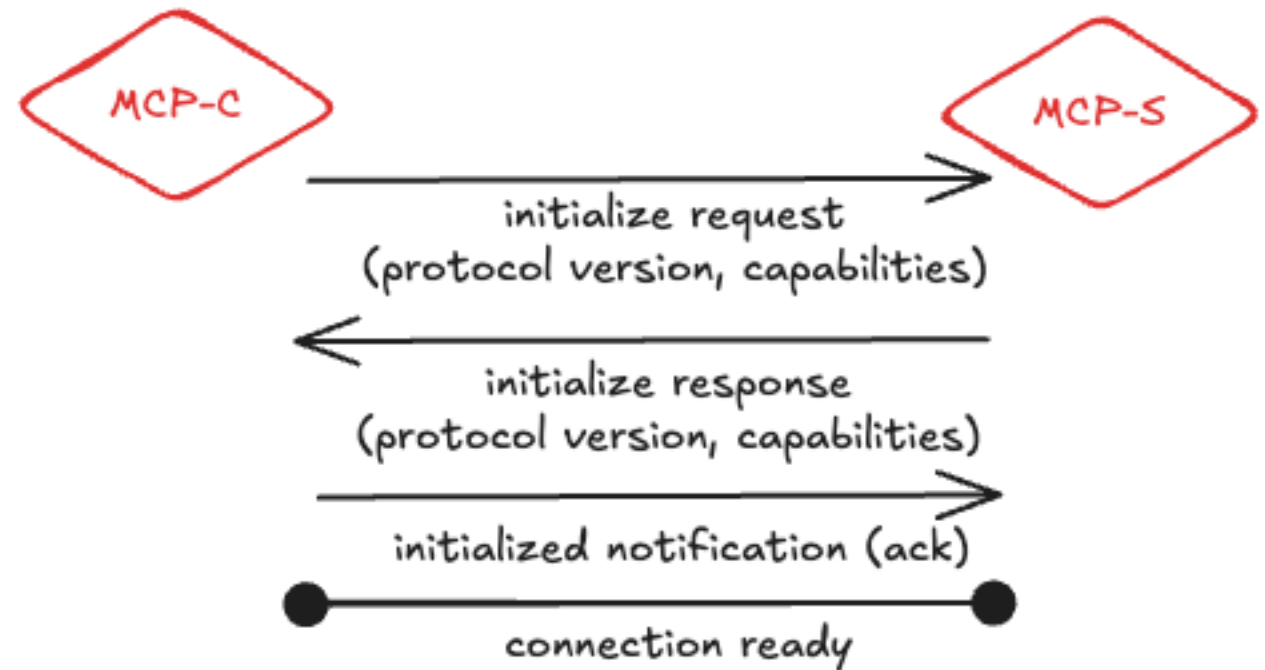
Tools Call Response:

```
event: message
data:
{"jsonrpc":"2.0","id":3,"result":{"content":
:[{"type":"text","text":"\nThis
Afternoon:\nTemperature: 74..F\nWind: 12
mph W\nForecast: Partly sunny. High near
74, with temperatures falling to around 72
in the afternoon"}}
```

MCP Discovery

MCP – Discovery

- When an agent needs to interact with a tool or service, MCP provides a resource discovery mechanism that lets MCP clients discover available resources
- The MCP client can use direct resources or resource templates
- You can read more about the resource discovery options at:
<https://modelcontextprotocol.io/docs/concepts/resources>
- The important thing to know is that the goal of resource discovery is to find out the following information:
 - Supported capabilities
 - Protocol versions
 - Custom metadata
- While there is no MCP server registry that MCP clients can search to dynamically discover all available MCP servers and their capabilities, there are MCP server directories:
 - <https://modelcontextprotocol.io/examples>



MCP-C to MCP-S request/response flow for the capabilities discovery

MCP Example: Resource Discovery

- The example is using the SQLite MCP Server that is running on the local machine:
<https://github.com/modelcontextprotocol/servers/tree/main/src/sqlite>
- Using Claude Desktop as the AI application with built-in MCP client functionality, the client can interact with the SQLite MCP server:

claude_desktop_config.json

```
"mcpServers": {  
  "sqlite": {  
    "command": "uvx",  
    "args": ["mcp-server-sqlite", "--db-path",  
      "/Users/shmcfarl/code/mcp-testing/sqlite/test.db"]  
  },  
}
```

Message exchange from using Claude to ask for a list of SQLite resources:

```
2025-04-09T18:08:37.964Z [sqlite] [info] Message from client:  
{ "method": "resources/list", "params": {}, "jsonrpc": "2.0", "id": 44 }  
2025-04-09T18:08:37.965Z [sqlite] [info] Message from server:  
{ "jsonrpc": "2.0", "id": 44, "result": { "resources": [ { "uri": "memo://insig  
hts", "name": "Business Insights Memo", "description": "A living  
document of discovered business  
insights", "mimeType": "text/plain" } ] } }
```

<https://modelcontextprotocol.io/docs/concepts/resources>

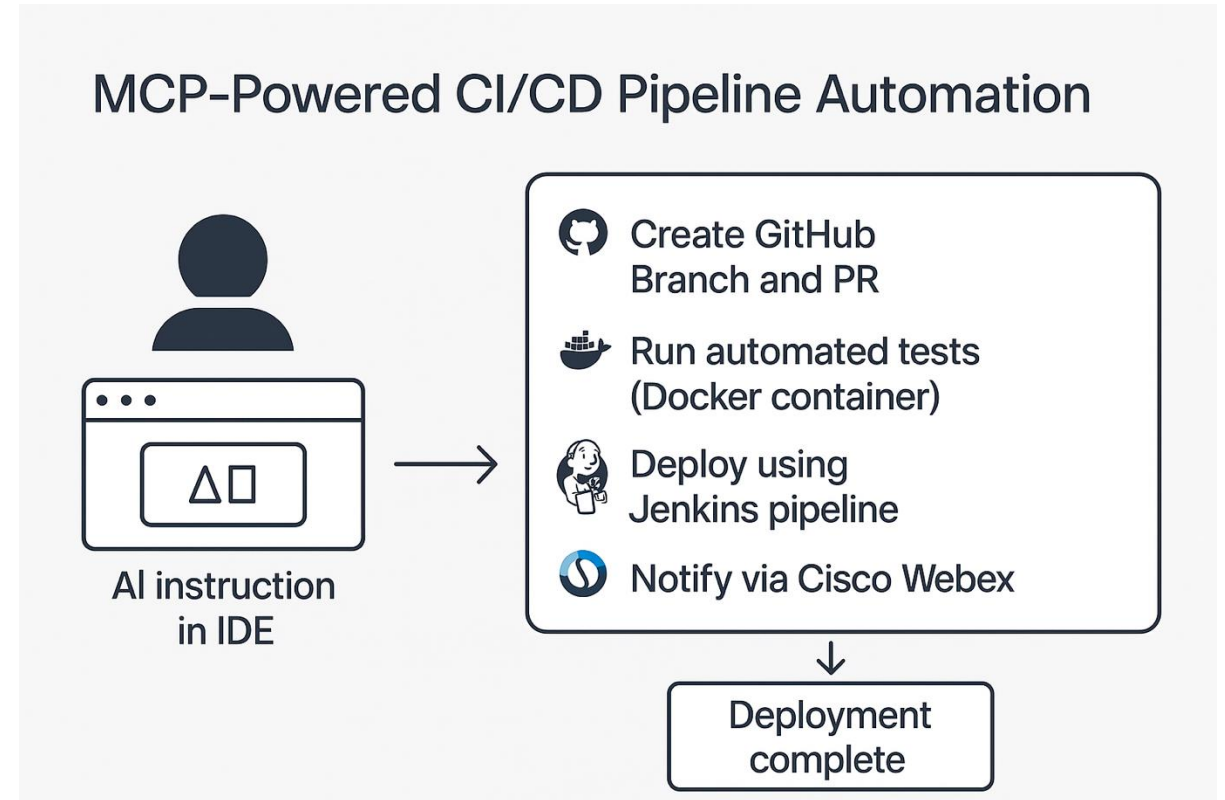
MCP Spec – resources/list

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

MCP Use Cases

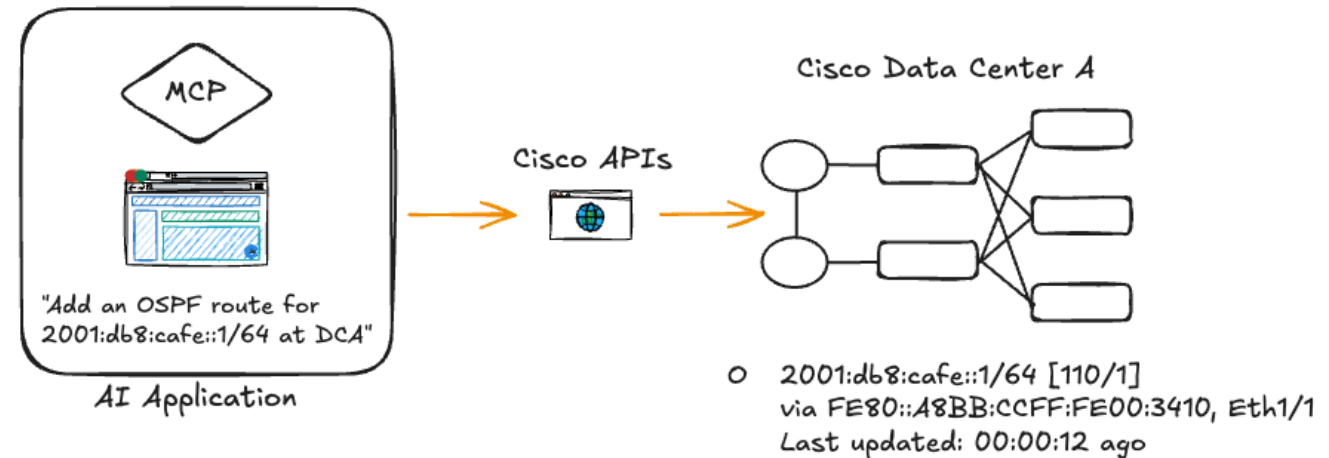
DevOps

- CI/CD Automation:
 - AI applications using MCP can automate entire CI/CD pipelines, seamlessly managing builds, tests, deployments, and notifications through Cisco Webex
- Efficient Code Management:
 - GitHub MCP integration enables an AI application or agent to manage branches, review pull requests, triage issues, and scan for vulnerabilities
- Infrastructure Automation:
 - With MCP Server integrations for Terraform and Ansible, your AI agent can quickly and accurately provision infrastructure or modify settings
- Streamlined Incident Response:
 - Cisco Webex integrated with MCP helps your AI application or agent actively engage in troubleshooting and incident management, significantly reducing response times



NetOps

- Dynamic Network Management:
 - MCP enables AI-driven management of network configurations using natural language through Cisco APIs or Infrastructure-as-Code (IaC) tools.
- Automated Network Monitoring:
 - With MCP, you can use an AI application or agent to monitor network performance, detect anomalies, and automatically remediate issues via Cisco solutions like ThousandEyes, Meraki Dashboard, and many more.
- Cloud Infrastructure Automation:
 - MCP allows you to use AI to manage cloud-based networking infrastructure, leveraging Kubernetes APIs and Cisco network controllers for intelligent automation.



SecOps

- Proactive Threat Response:

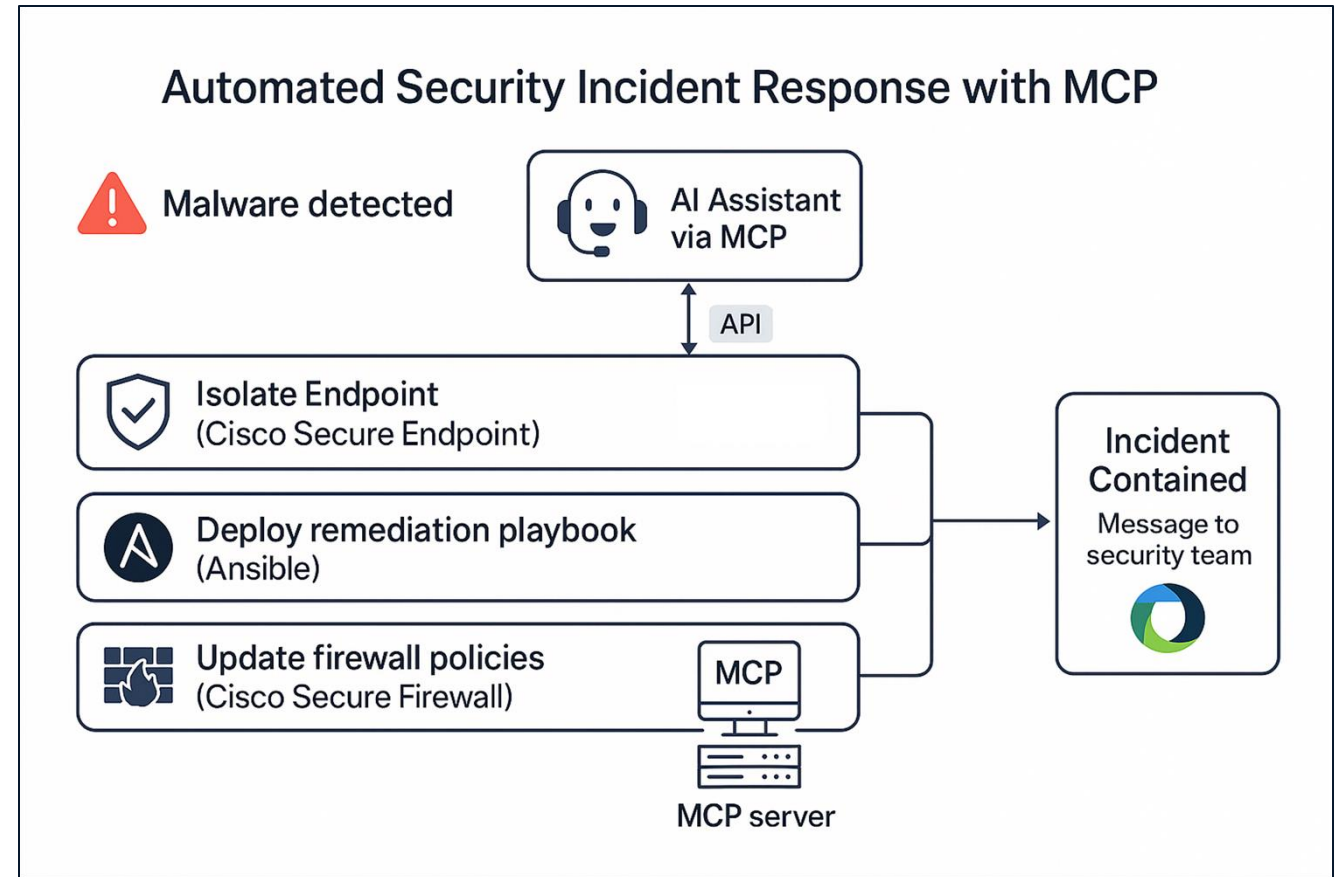
- AI agents using MCP swiftly detect and mitigate threats by adjusting firewall settings with Cisco Secure Firewall and automatically isolating compromised endpoints using Cisco Secure Endpoint.

- Automated Vulnerability Management:

- MCP integrations enable AI to identify vulnerabilities and generate immediate infrastructure or host configuration fixes through Ansible playbooks and Terraform providers.

- Real-time Incident Orchestration:

- With MCP, AI orchestrates comprehensive incident responses, isolating threats, deploying patches, and alerting teams via Cisco Webex.



Demo

References

Learning:

- MCP Architecture and Components Overview: https://youtu.be/Qdms0EHwhOw?si=VOGv4U97Y-S_VS1B
- MCP Use Cases: <https://youtu.be/NXFzZsvhJR8?si=mqQ6JqQqyPbQavPY>
- MCP Demo with NetBox Labs, GitHub, and more: https://youtu.be/fGQ_gI_RQ_M?si=qxdEtXib_sKjAl5v

Resources from the demo and presentation:

- MCP: <https://modelcontextprotocol.io/introduction>
- One of the many MCP server catalogs: <https://modelcontextprotocol.io/examples>
- Official GitHub MCP Server: <https://github.com/github/github-mcp-server>
- Official Argo CD MCP Server: <https://github.com/akuity/argocd-mcp>
- Official NetBox Labs MCP Server: <https://github.com/netboxlabs/netbox-mcp-server>
- Filesystem MCP Server: <https://github.com/modelcontextprotocol/servers/tree/main/src/filesystem>
- SQLite MCP Server: <https://github.com/modelcontextprotocol/servers-archived/tree/main/src/sqlite>

Conclusion

- MCP is off to a strong start, especially for DevOps teams experimenting with AI-driven automation
- MCP is still a young protocol – it gives you a clean foundation if you're building AI-enabled workflows that need to interact with infrastructure and tools
- MCP is ideal for:
 - Agents need to connect to multiple data sources and services in a standard way
 - It abstracts away the per-integration code complexity – just use the MCP SDK
 - You need it for a low toil platform or with IDE integrations
- What doesn't MCP do (at least today)?
 - MCP is not an agent-to-agent framework
 - MCP is not used for the creation, deployment, lifecycle management, and security of agents or tools
 - MCP is not an LLM
 - MCP is not a data source
 - MCP does not dynamically discover tools and services the MCP server will represent
- We also learned how MCP clients and servers interact with one another and over which types of protocol and messaging formats.

Thank you

CISCO Live !

