Cloud Contraptions
TRAINING & CONSULTING

To discuss this course and customizations:
Call: +1 434-509-6890 or Email: sales@cloudcontraptions.com

Model Context Protocol (MCP) for Developers

Class Duration

7 hours of live training delivered over 1-2 days to accommodate your scheduling needs.

Student Prerequisites

- Professional software development experience in TypeScript or Python
- Familiarity with REST APIs and JSON

Target Audience

Software engineers and platform engineers who want to extend AI coding assistants with internal tools, private data sources, and organizational APIs using MCP. Relevant for anyone building internal developer tooling on top of Claude, GitHub Copilot, or other MCP-compatible AI systems. Developers going deeper on the tool layer can continue with *Building Custom Tools and Skills for AI Coding Agents*, and those embedding MCP in autonomous systems can move to *Building AI Agents with Python and MCP*.

Description

The Model Context Protocol (MCP) is an open standard that defines how AI assistants connect to external tools and data sources. This course teaches developers how to both consume MCP servers (connecting your AI assistant to existing internal and third-party tools) and build them (exposing your organization's APIs and data to AI assistants safely and reliably). We cover the MCP specification, the server and client architecture, transport layers (stdio and streamable HTTP), tool definitions, resource endpoints, and prompt templates. Labs build working MCP servers in TypeScript and Python against realistic internal tool scenarios.

Learning Outcomes

- Explain the MCP specification: servers, clients, tools, resources, and prompts.

To discuss this course and customizations:
Call: +1 434-509-6890 or Email: sales@cloudcontraptions.com

- Connect an AI assistant (Claude, GitHub Copilot) to an existing MCP server.
- Build a working MCP server in TypeScript or Python that exposes tools and resources.
- Design safe tool schemas with appropriate input validation and error handling.
- Implement both stdio (local) and streamable HTTP (remote/hosted) transport layers.
- Apply authentication, authorization, and rate limiting to MCP server endpoints.
- Evaluate organizational patterns for sharing and governing internal MCP servers.

Training Materials

Comprehensive courseware is distributed online at the start of class. All students receive a downloadable MP4 recording of the training.

Software Requirements

Node.js 22+ or Python 3.12+, an MCP-compatible AI client (Claude Desktop, VS Code, Visual Studio, a JetBrains IDE, Zed, Cursor, or Windsurf), and Git.

Training Topics

MCP Fundamentals

- Why MCP: the context and tool integration problem
- MCP specification overview: servers, clients, hosts
- Transport layers: stdio vs. streamable HTTP (and the deprecated HTTP+SSE)
- MCP vs. function calling: key differences

Consuming MCP Servers

- Connecting MCP-aware hosts: Claude Desktop, VS Code, Visual Studio, JetBrains IDEs, Zed, Cursor, and Windsurf
- Browsing and invoking tools from an AI client
- Reading resources and using prompt templates
- Debugging connection and tool invocation issues

To discuss this course and customizations:
Call: +1 434-509-6890 or Email: sales@cloudcontraptions.com

Building MCP Servers in TypeScript

- Project setup with the MCP TypeScript SDK
- Defining tools: name, description, input schema
- Returning structured results and errors
- Exposing resources with URI templates

Building MCP Servers in Python

- Project setup with the MCP Python SDK
- Tool decorators and function handlers
- Async patterns for I/O-bound tools
- Resource and prompt template definitions

Authentication and Security

- MCP server authentication patterns
- OAuth 2.0 integration for hosted servers
- Input sanitization and safe execution
- Preventing prompt injection through tool outputs

Transport: Hosting and Distribution

- Local stdio servers: installation and config
- Remote streamable HTTP servers: deployment and TLS
- Packaging and distributing MCP servers to a team
- Registering servers in Claude Desktop, Copilot (VS Code/Visual Studio/JetBrains), Zed, Cursor, and Windsurf

MCP Registry: Public and Private

- The public MCP Registry (preview Sep 2025): canonical catalog and API for server discovery
- Public vs. private sub-registries for organizational isolation
- Server signing, verification, and trust scoring (incoming with the Registry GA)
- Upcoming spec evolutions: server-as-agent capabilities and registry maturation

Organizational Patterns

- Internal MCP registry for shared servers
- Governance: approving and versioning MCP servers
- Monitoring tool invocations in production

