ANTHROP\C

Claude Builder Club Agentic Masterclass 2

@Penn

October 29, 2025

Check-in for attendance!

Introductions

Today’s Presenter

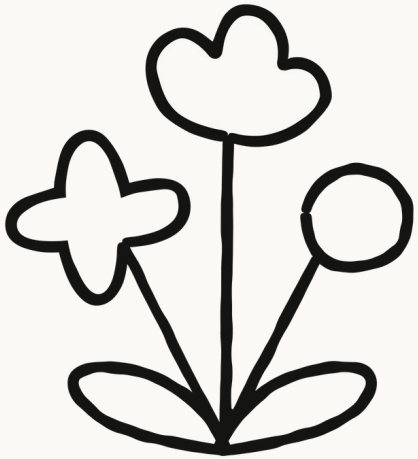
Albert Opher
Penn Claude Builder Club President
M&T | CIS + FNCE ‘25

Today’s Agenda: Agentic Development Deep Dive

- Claude Code CLI architecture & sandboxing
- Model Context Protocol (MCP) integrations
- Test-driven development workflows
- Research-first coding patterns
- Live demos: Real-world implementations

What is Claude Code?

Enterprise Agentic Architecture



- A Command Line, low-level, and un-opinionated Agentic coding tool developed by Anthropic
- Multi-model tool that draws on data from a code base
- **Technical Capabilities:**
    - Autonomous file system
    - Bash commands
    - Got operations
    - Test execution
    - Test Validation
    - Notebook Interaction

The NxM Integration Problem

- Complexity of integrating a set of N systems or applications with another set of M systems grows exponentially
- Creates a large number of custom, point-to-point connections
- Known software development issue
- A singular point of 1N to 1M is known as an integration point
- N commonly denotes LLM count and M denotes the connection points the LLMs attach to such as webhooks, apps, and controls
    - This doesn't surround LLM DL- this is high level abstraction
    - Causes redundant code and development time

Learn More Here:

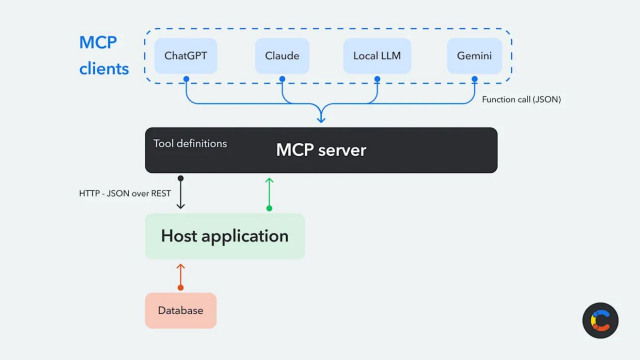
Solution? MCP!

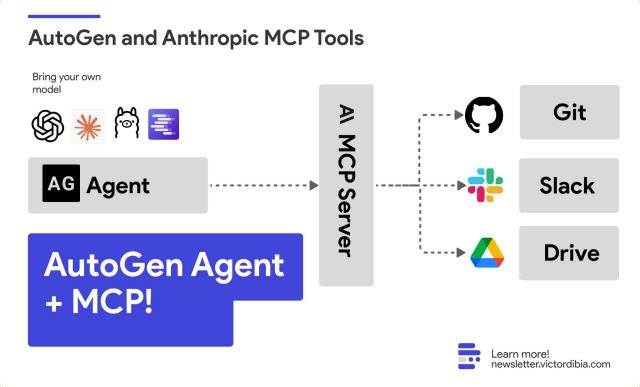
Model Context Protocol (MCP)

MCP provides a universal, open standard for connecting AI systems with data sources, replacing fragmented integrations with a single protocol

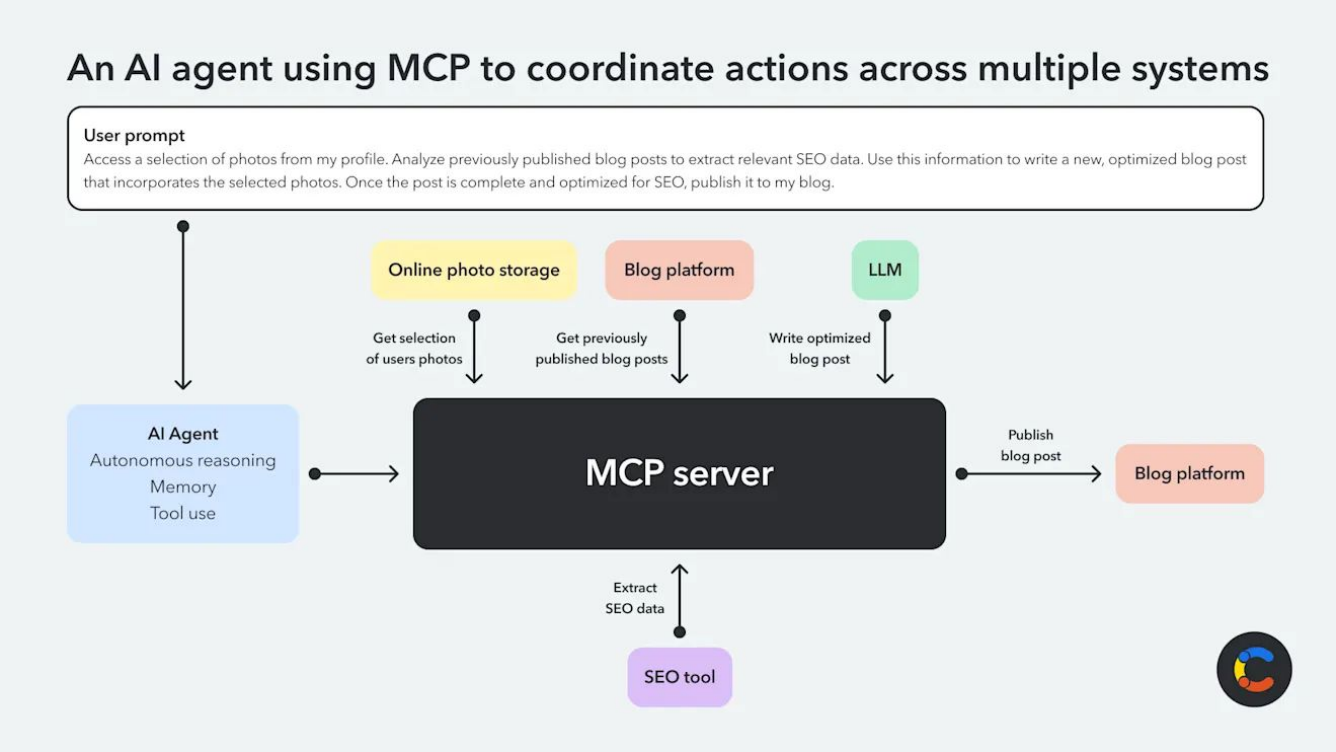
Framework:

1. **Client:** AI application (Claude Desktop, Claude Code)
2. **Server:** Data source connector (GitHub, Postgres, Slack, etc.)
3. **Transport:** JSON-RPC over stdio/HTTP/SSE
    a. JSON-RPC: lightweight, stateless remote procedure call for client-server interaction





Model Contect Protocol (MCP) II



1. Define the functionality with a tool schema
2. Build your MCP server
3. Add context handlers
4. Implement security
5. Register or share your MCP server

Architecture + Documentation

Research First Workflow + Claude Code Basics 1

Anthropic Recommended Prompting Structure:

Step 1: Research Phase
"Read the codebase structure and understand the current implementation patterns"

Step 2: Planning Phase
"Create a detailed implementation plan with:
- Files to modify
- Functions to add/change
- Test cases needed
- Potential edge cases"

Step 3: Implementation
"Execute the plan iteratively"

Step 4: Validation
"Run tests, verify outputs, iterate"

Claude Code Basics [Anthropic Research]:

Claude Code Basics 2

Claude Code Prompting:

 # 1. Write failing tests
"Create comprehensive unit tests for the user authentication module"

 # 2. Verify test failure
"Run pytest and confirm all tests fail"

 # 3. Commit tests
"Commit the test suite"

 # 4. Implement solution
"Now implement the authentication logic to make tests pass"
 # 5. Validate & iterate
"Run tests until all pass"

Add a Claude.md Markdown File to your Project Repo:

 # Project: [Name]

Architecture
[Key architectural decisions]

Code Style
[Conventions, patterns, linting rules]

Testing Strategy
[Test frameworks, coverage requirements]

Common Workflows
[Frequent tasks and their patterns]

External Dependencies
[APIs, services, environment variables]

Claude Code Basics 3

Adv Git Prompting:

 # 1. Create a Poll Request
"Create a pr named: polly"

 # 2. Create a Branch
"Create a br named: branchy"

 # 3. Fix PR comments
"In br titled branchy fix the comments on pr titled polly"

 # 4. Implement solution
"Now implement the authentication logic to make tests pass"
 # 5. Validate & iterate
"Run tests until all pass"

Building MCP Servers and Data Connectors:

1. **Define the functionality with a tool schema:** Decide on what functionality you want to make available to AI agents (fetching analytics, transcribing audio). Then write the tool definitions that describe these actions in a structured way.
2. **Build your MCP server:** Deploy a server that can host all of your tool definitions and provide endpoints that AI agents can access.
3. **Add context handlers:** If your app needs to provide relevant context to perform certain actions, add the required logic to provide that context when needed.
4. **Implement security:** Set up authentication, access controls, input validation, and human-in-the-loop approval where necessary.
5. **Register or share your MCP server:** Make it available through the MCP registry or share with your team.

Available SDKs: Python, TypeScript, C#, Java

Example Servers: GitHub, Slack, Postgres, Puppeteer, Google Drive, Stripe

Claude on the Cloud

Performance Benchmarks

Claude's Cloud:

Assign multiple coding tasks to Claude that run on Anthropic-managed cloud infrastructure. Each session runs in isolated environment with real-time progress tracking.

Claude in Industry:

Claude Sonnet 4.5 Results:

Edit capabilities: 9% error rate on Sonnet 4 to 0% on internal code editing benchmark.

Integration: Claude Sonnet 4.5 increased planning performance by 18% and end-to-end eval scores by 12%.

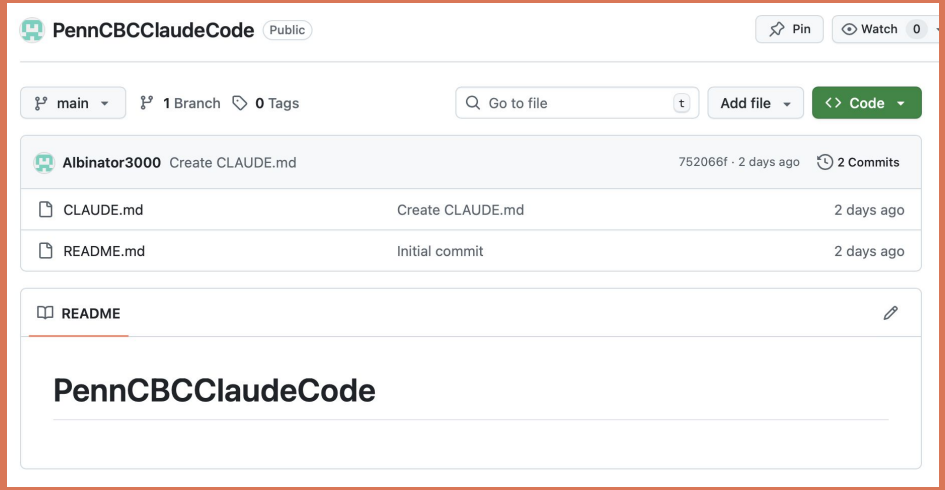
SWE-bench Verified: Opus 4.1: 74.5% on SWE-bench Verified without extended thinking.

Real-World Impact: Completed tasks in single pass that would normally take 45+ minutes of manual work. Hai security agents: 44% reduction in vulnerability intake time, 25% improvement in accuracy

Demo

What Improvements Can We Add to Our CBC Website?

penncbc.com



https://github.com/Albinator3000/PennCBCClaudeCode

YOU WIN FREE CLAUDE PRO

It's TRUE, for coming to this meeting, you have earned a free Claude Pro account.

To redeem:

- Fill out the form attached to the QR code
- Make sure you have a Claude.ai log-in that is connected to your **STUDENT EMAIL**

Note to students:
We will send out an email with your FREE Claude Pro login, this will take up to two business days!

Come to a meeting to get the QR code for Free Claude Pro

Q&A

ANTHROP\C

Thank you for coming!

ANTHROP\C