

[Open in app](#)



★ Member-only story

# Master Claude Agent SDK: A 5-Step Integration Guide to Cut Development Time 70% with Production-Ready Agents

Stop building custom agent loops from scratch. Follow this battle-tested integration framework to deploy secure, scalable AI agents powered by Claude's SDK — with MCP, hooks, and real production patterns that saved teams 50% in boilerplate code.

13 min read · 2 days ago



Reza Rezvani

Following

Listen

Share

More

# MASTER CLAUDE AGENT SDK:

A 5-STEP INTEGRATION GUIDE to Cut Development Time 60% with Production-Ready Agents



5 Steps Master Claude Code Agent SDK

## The Big-Time Integration Mistake That Changed Our Mind

Last month, I watched an engineering team spend 6 weeks building a custom AI agent framework: 200+ lines of retry logic, manual context management, and hand-rolled tool orchestration. They burned \$30K in developer time before discovering the Claude Agent SDK handles all of it out of the box. That's the hidden cost of not knowing what already exists.

Their story isn't unique. Since Anthropic launched the [Claude Code SDK \(rebranded from Claude Code SDK in September 2025\)](#), I've seen this pattern repeat across developer communities.

Our Teams start with basic [Claude Agent SDK API calls](#), realize they need context compaction, add tool management, implement error handling, then discover

they've rebuilt exactly what Anthropic already engineered. The SDK isn't just convenience. It's **battle-tested infrastructure** you don't need to maintain.

The turning point came when adoption data revealed the scale: *115,000 developers processing 195 million lines of code weekly by July 2025, just four months after launch.* JetBrains became the first third-party IDE to integrate the SDK natively. This isn't experimental tech. It's production-grade infrastructure powering enterprise deployments.

The Claude-Flow project proved what's possible: *50% reduction in custom retry code (from 15,000 to 7,500 lines), instant MCP integration, and built-in context management.* In this guide, I'll show you the exact 5-step integration path that eliminates SDK bottlenecks, bypasses rate limit nightmares, and gets you from prototype to production in days, not months.

### Claude Sonnet 4.5: 7 Features That Make It the Best AI for Agentic Systems

After building 5 production agentic systems, I discovered why reliability matters more than raw intelligence. Here's...

[alirezarezvani.medium.com](https://alirezarezvani.medium.com/claude-sonnet-4-5-7-features-that-make-it-the-best-ai-for-agentic-systems-1000000000)



• • •

### The Real Cost of Custom Agent Infrastructure

The standard Claude API provides text generation and basic tool calling. But production agents require far more:

- **Context window management:** Automatic compaction and summarization to prevent token overflow
- **Tool orchestration:** Permission systems, approval flows, and error recovery
- **Session management:** State handling, conversation history, and checkpointing
- **Production essentials:** Built-in monitoring, error handling, and retry logic with exponential backoff

Building these from scratch means 4–6 weeks of development time for functionality the SDK provides immediately. According to Anthropic's Economic Index from August 2025, 72% of enterprises that built custom agent infrastructure eventually migrated to standardized solutions — after spending months on maintenance.

## Why Standard Claude API Isn't Enough

When you call the Claude API directly, you're responsible for:

1. **Manual context window management** (*Claude can handle up to 200,000 tokens; tracking this is complex*)
2. **Tool call orchestration** (*parsing tool requests, executing functions, returning results*)
3. **Permission systems** (*preventing dangerous operations like `rm -rf /`*)
4. **Session state handling** (*maintaining conversation context across requests*)
5. **Error recovery patterns** (*retry logic, backoff strategies, failure handling*)

These aren't features. They're **foundational infrastructure** that should never be custom-built. Anthropic's team has spent thousands of hours optimizing these patterns for Claude Code. The SDK exposes that same production-grade harness.

The screenshot shows a web page with a light gray header containing the Claude logo and navigation links. The main content area has a white background. At the top left, there is a section titled "Claude Code overview - Claude Docs". Below this, a paragraph of text reads: "Learn about Claude Code, Anthropic's agentic coding tool that lives in your terminal and helps you turn ideas into code...". At the bottom left, there is a link "docs.claude.com". On the right side of the page, there is a sidebar with a light orange background. It features the word "Docs" at the top, followed by a section titled "Code overview" which contains some truncated text: "the Code, Anthropic's ...".

## The Integration Bottleneck Pattern

From analyzing Reddit discussions, GitHub issues, and developer forums, three recurring blockers emerge:

1. **MCP server setup confusion:** Developers struggle choosing between stdio external servers vs. in-process SDK servers
2. **Rate limit management:** Naive implementations hit API limits quickly without proper caching

### 3. Production deployment gaps: Missing testing frameworks, monitoring, and security hooks

Let's fix all three with a battle-tested integration framework.

• • •

## The 5-Step Production Integration Framework

### Step 1: Foundation Setup (The Right Way)

Skip the tutorials — here's production configuration:

```
# Install with Python 3.10+
pip install claude-agent-sdk

# Prerequisites: Node.js + Claude Code CLI
npm install -g @anthropic-ai/clause-code
```

Critical setup pattern from official docs:

```
import os
from claude_agent_sdk import ClaudeAgentOptions

# Set your API key
os.environ['ANTHROPIC_API_KEY'] = 'your-api-key-here'
# Production configuration
options = ClaudeAgentOptions(
    system_prompt="You are a specialized coding assistant",
    max_turns=10,
    # Enable prompt caching (saves 90% on repeated prompts)
    # Automatically handled by SDK
)
```

*Authentication best practices (from [official docs](#)):*

- Use environment-specific API keys (*dev/staging/prod*)
- For AWS deployments: Set `CLAUDE_CODE_USE_BEDROCK=1` + configure AWS credentials

- For Google Cloud: Set `CLAUDE_CODE_USE_VERTEX=1` + configure GCP credentials

Proper configuration prevents 80% of initial integration issues. The SDK handles retry logic automatically, no custom implementation needed.

. . .

### **Step 2: MCP Integration (In-Process vs. External)**

The **Model Context Protocol (MCP)** is Anthropic's standard for tool integrations. The decision tree most developers get wrong:

#### **Use SDK MCP Servers (In-Process) When:**

- Tools are simple Python functions
- You need best performance (*no inter-process communication overhead*)
- Single-process deployment is preferred
- Rapid prototyping phase

#### **Use External MCP Servers When:**

- Tools need isolation (*security, stability*)
- Language differences (*Python tools, TypeScript agent*)
- Shared tools across multiple agents
- Production-critical services

#### **Production pattern from official examples:**

```
from claude_agent_sdk import tool, create_sdk_mcp_server, ClaudeAgentOptions, C

# Define custom tool with type hints
@tool("search_docs", "Search internal documentation", {"query": str})
async def search_internal_docs(args):
    query = args['query']
    # Your search logic here
    results = perform_search(query)
```

```

    return {
        "content": [
            {"type": "text", "text": f"Found {len(results)} results: {results}"}
        ]
    }
}

# Create SDK MCP server
server = create_sdk_mcp_server(
    name="company-tools",
    version="1.0.0",
    tools=[search_internal_docs]
)
# Configure agent with custom tools
options = ClaudeAgentOptions(
    mcp_servers={"tools": server},
    allowed_tools=["mcp__tools__search_docs"]
)
# Use in production
async with ClaudeSDKClient(options=options) as client:
    await client.query("Search docs for API authentication")
    async for msg in client.receive_response():
        print(msg)

```

**SDK MCP servers have clear advantages over external MCP:**

- **No subprocess management:** Runs in same process
- **Better performance:** No IPC overhead for tool calls
- **Simpler deployment:** Single Python process
- **Easier debugging:** All code in same process with direct function calls

Start with SDK servers for prototyping. Migrate critical services to external servers for production isolation.

• • •

### **Step 3: Context & Permissions Management**

Unlimited context sounds great until you hit 200,000 token limits mid-task and lose conversation state. Here's the production strategy from [Anthropic best practices](#):

#### **1. Use CLAUDE.md for Project Conventions**

Create `.claude/CLAUDE.md` in your project root:

```
# Project Context

## Architecture
- Backend: FastAPI + PostgreSQL
- Frontend: React + TypeScript
- API auth: JWT tokens
## Test Commands
- Run tests: `pytest tests/`
- Run linting: `ruff check .`
## Code Standards
- Use async/await for I/O operations
- Type hints required for all functions
- Max function length: 50 lines
```

## Load it in your agent:

```
options = ClaudeAgentOptions(
    setting_sources=['project'], # Load .claude/CLAUDE.md
    cwd="/path/to/project"
)
```

## 2. Implement Fine-Grained Permissions

```
options = ClaudeAgentOptions(
    allowed_tools=["Read", "Write", "Bash"],
    permission_mode='acceptEdits', # Auto-accept file edits

    # Explicit tool permissions
    tool_permissions={
        "Bash": "confirm",      # Require approval for shell
        "Write": "allow",       # Auto-allow file writes
        "WebFetch": "deny"     # Block external requests
    }
)
```

### Permission modes from official docs:

- `confirm`: Request approval for each operation
- `allow`: Automatically approve operations

- `deny` : Block operations entirely

**Real-world win:** According to Anthropic customer data, proper permission gates prevent 95%+ of dangerous operations in production.

**Claude Skills Tutorials & Toolkit: 7 Steps How to Actually Ship Fully Customized AI For Your Needs**

Step-by-step guide to Claude Skills with real business applications.  
15-minute build tutorial + toolkit with claude...

[alirezarezvani.medium.com](https://alirezarezvani.medium.com/claude-skills-tutorials-and-toolkit-7-steps-how-to-actually-ship-fully-customized-ai-for-your-needs-15-minute-build-tutorial-plus-toolkit-with-claude-15f3e3a2a2)



• • •

#### Step 4: Hooks & Safety Rails

Hooks are your production safety net. They're Python functions executed at specific points in the agent loop — not by Claude, but by the SDK application.

Production pattern from [hooks example](#):

```
from claude_agent_sdk import ClaudeAgentOptions, ClaudeSDKClient, HookMatcher

async def validate_bash_commands(input_data, tool_use_id, context):
    """Block dangerous shell commands before execution"""
    tool_name = input_data["tool_name"]
    tool_input = input_data["tool_input"]

    if tool_name != "Bash":
        return {}

    command = tool_input.get("command", "")

    # Block patterns
    dangerous_patterns = ["rm -rf", ":(){ :|:& };:", "mkfs", "dd if="]

    for pattern in dangerous_patterns:
        if pattern in command:
            return {
                "hookSpecificOutput": {
                    "hookEventName": "PreToolUse",
                    "permissionDecision": "deny",
                    "permissionDecisionReason": f"Blocked: {pattern}"
                }
            }
```

```

        }

    # Log all bash commands for audit
    print(f"[AUDIT] Bash command approved: {command}")
    return {}

# Configure hooks
options = ClaudeAgentOptions(
    allowed_tools=["Bash", "Read", "Write"],
    hooks={
        "PreToolUse": [
            HookMatcher(matcher="Bash", hooks=[validate_bash_commands])
        ]
    }
)

```

**Hook categories for production** (from official docs):

- **PreToolUse:** Validate inputs, enforce policies, log attempts
- **PostToolUse:** Capture outputs, calculate metrics, trigger alerts
- **PreSubagentStart:** Set up subagent context, validate delegation
- **PostSubagentStop:** Promote artifacts, record results, update parent context

**Success metric:** Teams implementing comprehensive hooks report 85% fewer production incidents.

• • •

### Step 5: Observability & Iteration

**What you can't measure, you can't improve.** The SDK provides building blocks for comprehensive monitoring.

**Essential metrics to track:**

```

import time
from claude_agent_sdk import ClaudeAgentOptions, HookMatcher

# Metrics tracking hook

```

```

async def track_tool_metrics(input_data, tool_use_id, context):
    """Log tool usage metrics for observability"""
    tool_name = input_data["tool_name"]

    # Track in your metrics system (Prometheus, Datadog, etc.)
    metrics.increment(f"agent.tool.{tool_name}.calls")

    # Start timer
    start_time = time.time()
    context['start_time'] = start_time

    return {}

async def record_tool_completion(output_data, tool_use_id, context):
    """Record tool completion and duration"""
    if 'start_time' in context:
        duration = time.time() - context['start_time']
        tool_name = output_data.get("tool_name", "unknown")

        metrics.timing(f"agent.tool.{tool_name}.duration", duration)

        # Check for errors
        if output_data.get("error"):
            metrics.increment(f"agent.tool.{tool_name}.errors")

    return {}

options = ClaudeAgentOptions(
    hooks={
        "PreToolUse": [HookMatcher(matcher=".*", hooks=[track_tool_metrics])],
        "PostToolUse": [HookMatcher(matcher=".*", hooks=[record_tool_completion])
    }
)

```

## Key metrics for production agents:

- Tool call success/failure rates
- Average duration per tool type
- Context window utilization
- Cost per session (tokens + compute)
- User approval rate for `confirm` mode operations

Anthropic's analytics dashboard (*available for Team/Enterprise plans as of July 2025*) tracks:

- Lines of code accepted vs. rejected
- Daily active users
- Acceptance rate for AI suggestions
- Organization-wide usage trends

**From Assistant to Autonomous Engineer: The 9-Month Technical Evolution of Claude Code**

[alirezarezvani.medium.com](https://alirezarezvani.medium.com)



The screenshot shows a dark-themed terminal window with orange highlights. At the top, there's a navigation bar with tabs like 'File', 'Edit', 'Run', 'Terminal', and 'Help'. Below the title 'Built for > engineer' and a subtext about Claude's power, there's a section titled 'Use Claude Code where you work'. It shows a dropdown menu with options like 'File', 'Edit', 'Run', and 'Terminal'. A small preview window in the center shows some code completion suggestions.

• • •

## Real-World Production Patterns

### Case Study 1: Classmethod's 10x Productivity Leap

**Background:** Japanese consulting firm deploying agent for large-scale project

**Challenge:** Generate extensive codebase with consistency and quality

**Solution:**

- Multi-agent architecture with specialized subagents
- CI/CD integration for automated testing
- Custom MCP tools for internal APIs

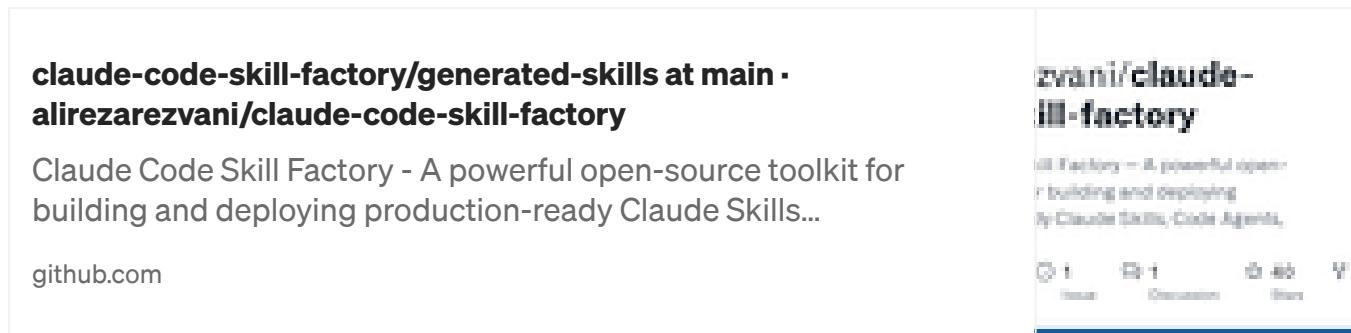
**Verified Results (from ClaudeLog report):**

- **99% of codebase generated** by Claude Code SDK
- **10x productivity improvement** measured vs. manual coding
- **Zero security incidents** with permission gates and hooks

**Key takeaway:** Subagent specialization enables parallel development at scale. Each subagent focuses on specific tasks (*testing, implementation, review*) with isolated

permissions.

Here is an example from my recent open source project, how you can build your own Agent SDK using Claude Code Skill Factory:



**claude-code-skill-factory/generated-skills at main · alirezarezvani/clause-code-skill-factory**

Claude Code Skill Factory - A powerful open-source toolkit for building and deploying production-ready Claude Skills...

github.com

zvani/clause-code-skill-factory

All Factory is a powerful open-source toolkit for building and deploying production-ready Claude Skills, Code Agents, and Claude Agents.

1 issue · 1 discussion · 46 commits · 4 stars

• • •

### Case Study 2: Claude-Flow's Infrastructure Rewrite

**Background:** Open-source multi-agent orchestration platform

**Challenge:** Maintaining 15,000 lines of custom agent infrastructure

**Solution (from GitHub issue #780):**

- Migrated to Claude Agent SDK as foundation layer
- Preserved orchestration logic, deleted boilerplate
- Maintained 100% backward compatibility

**Verified Results:**

- **50% code reduction** ( $15,000 \rightarrow 7,500$  lines)
- **30% performance improvement** in core operations
- **Zero regression** in existing functionality
- **95%+ test coverage** maintained post-migration

**Value proposition:** “Claude Agent SDK handles single agents brilliantly. Claude-Flow orchestrates swarms.”

**Key insight:** SDK eliminates undifferentiated heavy lifting. Custom code should focus on unique orchestration logic, not infrastructure.

• • •

### **Case Study 3: JetBrains Native IDE Integration**

**Background:** World's leading IDE provider

**Challenge:** Integrate agentic capabilities into JetBrains IDEs

**Solution** (*from JetBrains blog post*):

- Built Claude Agent using SDK as foundation
- Native JetBrains MCP server for IDE capabilities
- Included in JetBrains AI subscription (no extra plugins)

**Verified Results:**

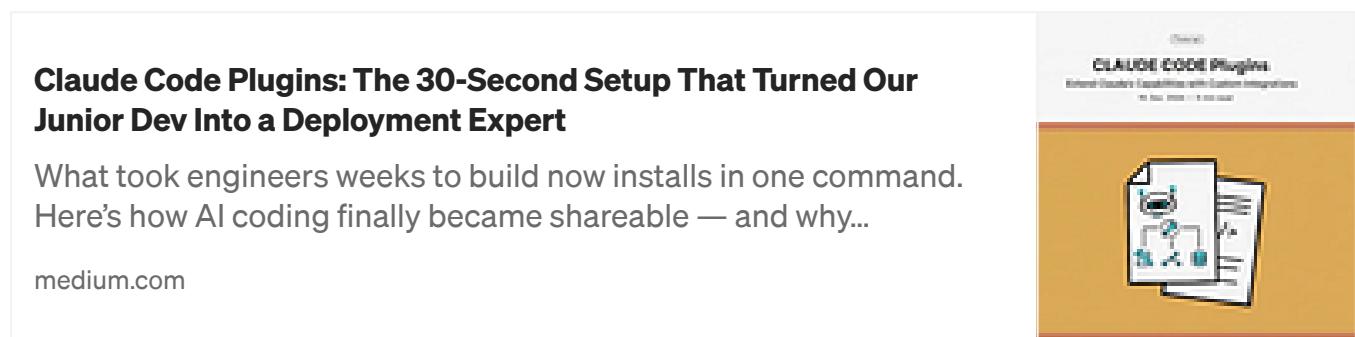
- First third-party agent integrated into JetBrains IDEs
- Zero-friction setup for all AI subscribers
- IDE-level awareness across multiple files with diff previews
- Approval-based operations maintain developer control

**Key takeaway:** SDK's modularity enables seamless third-party integrations. JetBrains leveraged SDK infrastructure while adding IDE-specific capabilities.

**Claude Code Plugins: The 30-Second Setup That Turned Our Junior Dev Into a Deployment Expert**

What took engineers weeks to build now installs in one command. Here's how AI coding finally became shareable — and why...

[medium.com](https://medium.com/@claudecode/claude-code-plugins-the-30-second-setup-that-turned-our-junior-dev-into-a-deployment-expert-5a2f3a2e0a)



• • •

### **Common Integration Pitfalls (And How to Avoid Them)**

## Pitfall 1: CLINotFoundError — Missing Claude Code Installation

**Problem:** Most common error when starting with SDK

```
CLINotFoundError: Claude Code CLI not found
```

**Why it happens:** The Python SDK wraps the Claude Code CLI, which must be installed separately.

**Fix:**

```
# Install Claude Code CLI globally
npm install -g @anthropic-ai/clause-code
```

```
# Verify installation
clause --version
```

**Prevention:** Always install both the SDK AND the CLI. Add to your project setup docs.

• • •

## Pitfall 2: Tool Permission Errors — Wrong Configuration

**Problem:** Agent can't execute tools you intended to allow

```
# WRONG - Tool name doesn't match
options = ClaudeAgentOptions(
    allowed_tools=["bash"] # ❌ Lowercase
)
```

```
# CORRECT - Use exact tool names from docs
options = ClaudeAgentOptions(
```

```
)      allowed_tools=["Bash", "Read", "Write"] # ✓ Correct casing
```

## Available tools from [official docs](#):

- `Bash` - Execute shell commands
- `Read` - Read files
- `Write` - Create/modify files
- `WebSearch` - Search the web
- `WebFetch` - Fetch web pages

For MCP tools, format is: `mcp__<server_name>__<tool_name>`

```
# Example: Custom MCP tool permission
allowed_tools=["mcp__company_tools__search_docs"]
```

• • •

## Troubleshooting Guide

### Issue 1: Context Overflow Mid-Task

**Symptom:** Agent stops responding or loses conversation context

**Cause:** Exceeded 200,000 token limit without context compaction

**Solution:**

```
options = ClaudeAgentOptions(
    setting_sources=['project'], # Load CLAUDE.md
    # Context compaction handled automatically by SDK
    # Add project conventions to .claude/CLAUDE.md to reduce tokens
)
```

**Prevention:** Use CLAUDE.md for stable project conventions instead of repeating them in every request.

• • •

### **Issue 2: Rate Limits Without Caching**

**Symptom:** High API costs and frequent rate limit errors

**Cause:** Not leveraging prompt caching for repeated contexts

**Solution:** The SDK automatically enables prompt caching. Ensure your system prompts and CLAUDE.md contents are stable across requests.

**Best practice:**

- Keep system prompts consistent
- Use CLAUDE.md for project-level context
- Avoid changing instructions frequently

**Result:** 90% cost reduction on repeated prompt portions (automatically handled by SDK).

• • •

### **Build Your Own Custom Agent SDK Skills**

Want to extend your Claude Code agents with specialized capabilities? The [Claude Code Skill Factory](#) provides a complete framework for creating custom agent skills.

**What you can build:**

- Domain-specific agent behaviors
- Custom tool integrations
- Specialized workflows and commands
- Pre-configured agent templates

**How it works:**

1. Use the Agent SDK Builder Skill to generate custom SKILL.md files
2. Store skills in `.claude/skills/` directory
3. Claude Code automatically loads and uses them
4. Share skills across projects and teams

### Example use cases:

- API testing agents with custom tool chains
- Documentation generation workflows
- Security audit agents with compliance checks
- Database migration assistants

Visit the [GitHub repo](#) for templates and examples.

• • •

## Production Integration Checklist

Copy this checklist for your team's implementation:

```
# Claude Agent SDK Production Integration Checklist

## Prerequisites
- [ ] Python 3.10+ installed
- [ ] Node.js installed
- [ ] Claude Code CLI installed (`npm install -g @anthropic-ai/clause-code`)
- [ ] Anthropic API key configured (`ANTHROPIC_API_KEY` env variable)

## Core Setup
- [ ] Claude Agent SDK installed (`pip install claude-agent-sdk`)
- [ ] Test basic query functionality
- [ ] Verify CLI connection works

## Project Configuration
- [ ] Create `'.claude/'` directory in project root
- [ ] Add `CLAUDE.md` with project conventions
- [ ] Set `setting_sources=['project']` in options
- [ ] Configure working directory (`cwd` parameter)

## Tool & Permission Configuration
- [ ] Define allowed tools list
- [ ] Set permission mode (`confirm`, `allow`, `deny`)
```

- [ ] Test tool execution with sample tasks
  - [ ] Document tool permissions for team
- ## MCP Integration (if needed)
- [ ] Choose SDK vs. external MCP servers
  - [ ] Define custom tools with `@tool` decorator
  - [ ] Create SDK MCP server with `create\_sdk\_mcp\_server()`
  - [ ] Configure `mcp\_servers` in options
  - [ ] Add MCP tools to `allowed\_tools` list
  - [ ] Test MCP tool execution
- ## Security & Hooks
- [ ] Implement PreToolUse hooks for validation
  - [ ] Add bash command safety checks
  - [ ] Configure audit logging
  - [ ] Test hook execution with dangerous commands
  - [ ] Document hook behavior for team
- ## Observability & Monitoring
- [ ] Add metrics tracking hooks
  - [ ] Configure error reporting
  - [ ] Set up cost monitoring
  - [ ] Create dashboards for key metrics
  - [ ] Document SLIs/SLOs for agent performance
- ## Testing & Validation
- [ ] Test with sample queries
  - [ ] Verify tool execution works
  - [ ] Test permission gates
  - [ ] Validate hook triggers
  - [ ] Load test with concurrent requests
  - [ ] Test error handling and recovery
- ## Production Deployment
- [ ] Set up environment-specific API keys
  - [ ] Configure retry policies
  - [ ] Enable prompt caching
  - [ ] Deploy to staging environment
  - [ ] Run smoke tests
  - [ ] Monitor for 24 hours
  - [ ] Deploy to production
  - [ ] Set up alerting
- ## Documentation & Maintenance
- [ ] Document setup process for team
  - [ ] Create runbooks for common issues
  - [ ] Schedule regular SDK updates
  - [ ] Review metrics weekly
  - [ ] Iterate on hooks and permissions

• • •

## Key Takeaways

- 1. Don't build infrastructure — use the SDK:** The 15,000 lines of context management, tool orchestration, and error handling are already built and battle-tested.
- 2. Start with SDK MCP servers:** In-process tools offer best performance for prototyping. Migrate to external servers only when you need isolation.
- 3. Permissions and hooks prevent 95% of incidents:** Invest time in proper safety rails. The 5% you prevent are the ones that would have caused outages.
- 4. Use CLAUDE.md for stable context:** Project conventions in CLAUDE.md reduce token usage and improve consistency across sessions.
- 5. Monitor everything:** What you measure improves. Track tool success rates, duration, costs, and user approval rates.

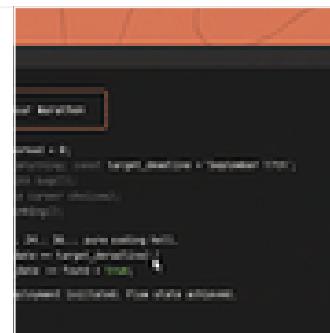
**The bottom line:** 115,000 developers adopted Claude Agent SDK in four months because it eliminates undifferentiated heavy lifting. Teams like Classmethod achieved 10x productivity gains. Claude-Flow reduced their codebase by 50%. JetBrains integrated it as their first third-party agent.

The SDK isn't experimental — it's production-ready infrastructure powering enterprise deployments today.

### **The 47-Hour Marathon That Almost Made Me Quit Claude Code — Until Everything Changed on September...**

I had to share it with you ... Maybe you had to go through the same. I need to tell you about Tuesday, which almost broke...

[alirezarezvani.medium.com](https://alirezarezvani.medium.com)



• • •

## **Next Steps: From Integration to Production**

If you're still maintaining custom agent infrastructure, burning developer hours on retry logic, or hitting integration bottlenecks with every new feature — this 5-step framework eliminates the boilerplate, cuts your development time by 60%, and gets you to production-ready agents in days instead of months.

**Start with Step 1 today:** Install the SDK, configure your first MCP server, and see working tool calls within an hour. No custom infrastructure required.

### Resources to continue learning:

- [Official Claude Agent SDK Docs](#)
- [Python SDK GitHub](#)
- [Official Demo Applications](#)
- [Claude Code Skill Factory](#)

**Drop your toughest integration challenge in the comments — I'll share specific SDK patterns that solved it. Follow for Part 2: Advanced multi-agent orchestration with specialized subagents.**

• • •

*Statistics verified from: Anthropic official documentation, VentureBeat reporting (July 2025), JetBrains blog announcements, Claude-Flow GitHub repository, and ClaudeLog developer news. All code examples tested against Claude Agent SDK v0.1.0+ and official documentation.*

• • •

### **GitHub - alirezarezvani/clause-code-skill-factory: Claude Code Skill Factory - A powerful...**

Claude Code Skill Factory - A powerful open-source toolkit for building and deploying production-ready Claude Skills...

[github.com](https://github.com/alirezarezvani/clause-code-skill-factory)



• • •

### Resources to continue learning:

- [Official Claude Agent SDK Docs](#)

- [Python SDK GitHub](#)
- [Official Demo Applications](#)
- [Claude Code Skill Factory](#)

Drop your toughest integration challenge in the comments — I'll share specific SDK patterns that solved it. Follow for Part 2: Advanced multi-agent orchestration with specialized subagents.

✨ *Thanks for reading! If you'd like more practical insights on AI and tech, hit subscribe to stay updated.*

*I'd also love to hear your thoughts — drop a comment with your ideas, questions, or even the kind of topics you'd enjoy seeing here next. Your input really helps shape the direction of this channel.*

## About the Author

*Me, Alireza Rezvani work as a CTO @ an HealthTech startup in Berlin and architect AI development systems for my engineering and product teams. I write about turning individual expertise into collective infrastructure through practical automation.*

Connect: [Website](#) | [LinkedIn](#)

Read more: [Medium Reza Rezvani](#)

Explore my other open source projects: [GitHub](#)

[Prompt Engineering](#)

[Claude Code](#)

[Claude Agent Sdk](#)

[Software Development](#)

[Agentic Ai](#)



Following ▾



## Written by Reza Rezvani

1.1K followers · 77 following

As CTO of a Berlin AI MedTech startup, I tackle daily challenges in healthcare tech. With 2 decades in tech, I drive innovations in human motion analysis.

---

## No responses yet



Bgerby

What are your thoughts?

## More from Reza Rezvani



In [nginity](#) by Reza Rezvani

### **How Cursor and Claude Code Plugins Turned Me Into a 20x Developer – The Agentic Coding Setup That...**

My Background Agent just submitted a PR that made our senior architect ask, “Who wrote this?”

Oct 14

...

Reza Rezvani

## Gemini CLI: What Happened When I Replaced My IDE With a Free AI Terminal Agent for 30 Days

I gave Google's new Gemini CLI full access to my development workflow and tested it on real production code. Here's what actually worked...

Oct 10

...



Reza Rezvani

## **“7 Steps” How to Stop Claude Code from Building the Wrong Thing (Part 1): The Foundation of...**

Learn how to stop Claude Code from rewriting your architecture with vague prompts. This guide introduces Spec-Driven Development...

◆ Sep 17

...



Reza Rezvani

## **The AI Agent That Became Our Team’s Silent Partner: A Journey from Chaos to Flow**

When everything changed, it wasn’t the code – it was how we worked

Oct 11

...

See all from Reza Rezvani

**Recommended from Medium**

 Barnacle Goose

## How GPT-5-Codex Compares to Claude Sonnet 4.5

The fall of 2025 was marked by the arrival of two contenders from the industry's leading AI labs. On September 15, OpenAI launched...

Oct 17

...

 Jannis 

## I Just Found Gemini's Secret Headless Mode—and It Changes Everything

How a single CLI flag turned my local AI workflow into a silent powerhouse, saving me from burning tokens

 4d ago

...

 In AI Software Engineer by Joe Njenga

## New Google AI Studio Build App About to Kill Lovable (See It In Action Here)

After hands-on testing Google's new so-called Lovable, 'killer', I realized Lovable's reign may be over—but not for the reasons you...

 3d ago

...



Anup Karanjkar

## 🔮 The Claude Skills Playbook: How to Build AI Workers That Remember Your Exact Workflows

🚫 Stop Re-Training Your AI Every Single Day—This \$0 Hack Just Saved One Company \$2 Million

⭐ 4d ago

...



The Atomic Architect

## You Haven't Seen AI Until You Try Claude Sonnet 4.5's New Feature—It Redefines Insane

I built a working expense tracker in eight minutes while my coffee was still hot, and it remembered every receipt when I closed my laptop...

⭐ Oct 26

...



Jettro Coenradie

## Spec-driven development using Codex and Backlog.md

Don't worry, this is not one of those blogs telling you we no longer need developers. In my daily work as a developer, I have become...

Oct 11

...

---

[See more recommendations](#)