Building AI Agents with the Claude Agent SDK

FROM FUNDAMENTALS TO PRODUCTION DEPLOYMENTS

- Autonomous Tool Execution
- Context Management
- Multi-Agent Orchestration
- Secure by Design

COMPLETE EXAMPLES IN

Python | TypeScript

DESIGN PATTERNS | SECURITY PRACTICES | OPERATIONAL STRATEGIES

YURIY ALEKSOV

Building AI Agents with the Claude Agent SDK

From Fundamentals to Production Deployments

Steve T. Publications

This book is available at
https://leanpub.com/buildingaiagentswiththeclaudeagentsdk

This version was published on 2026-07-13

Contents

From Fundamentals to Production Deployments

About this book: The Claude Agent SDK transforms Anthropic's Claude Code agent harness into a programmable library, giving developers access to the same autonomous tool execution, context management, and multi-agent orchestration that powers one of the most capable AI coding tools in existence. This book takes you from the fundamentals of AI agent architecture through advanced production patterns, with complete code examples in both Python and TypeScript. Whether you are building internal developer tools, customer-facing automation, or enterprise-grade multi-agent systems, you will learn the design patterns, security practices, and operational strategies that separate prototypes from production.

Introduction: The Agent Revolution

What Are AI Agents and Why They Matter

In late 2023, a developer asked ChatGPT to fix a bug in their Python application. The model replied with corrected code, and the developer copy-pasted it into their editor. This was the standard interaction pattern: human asks, model answers, human acts.

By mid-2025, that same developer could ask an AI agent to find the bug, read the relevant files, run tests, apply the fix, and commit the change – all without leaving the terminal. The model did not just answer; it acted. It used tools, iterated on failures, and produced a result rather than a suggestion.

This is the fundamental shift that defines AI agents: they do not merely generate text in response to prompts. They perceive their environment through tools, reason about what to do next, execute actions, observe results, and iterate until a task is complete. The difference between a chatbot and an agent is the difference between a consultant who writes you a memo and a colleague who actually does the work.

The importance of this shift cannot be overstated. Agents transform AI from a productivity multiplier into an autonomous workforce component. They handle repetitive multi-step workflows, synthesize information from disparate sources, execute code changes with verification, and escalate to humans only when they encounter genuine ambiguity. The economic implications are profound: tasks that previously required hours of human coordination can now be completed in minutes by coordinated agent systems.

But building agents well is hard. The naive approach – sending a prompt to an LLM and hoping it figures out the rest – produces unreliable, unpredictable results. Production agents require careful architecture: tool definitions that guide rather than constrain, permission systems that protect without paralyzing, memory systems that retain context without bloating costs, and observability pipelines that make debugging possible in systems where the model makes its own decisions.

From Claude Code CLI to Agent SDK

The Claude Agent SDK did not emerge from a greenfield research project. It evolved from Claude Code, Anthropic's AI-powered coding assistant that debuted in 2025 as a command-line tool for developers. Claude Code was impressive because it could navigate codebases, edit files, run tests, and fix bugs with a level of autonomy that exceeded earlier coding assistants. But its capabilities were locked behind a CLI interface, accessible only to developers sitting at their terminals.

Anthropic recognized that the agent harness powering Claude Code – the loop that reads Claude's tool-use requests, executes them, feeds results back, and repeats until completion – was itself a valuable product. In late 2025, they extracted this harness into a library called the "Claude Code SDK." By early 2026, recognizing that its applications extended far beyond coding, they renamed it the Claude Agent SDK.

The result is a Python and TypeScript library that gives you programmatic access to the same capabilities that power Claude Code: built-in file operations, shell command execution, web search, code discovery, structured tool use, lifecycle hooks, subagent spawning, and session management. You get all of this without implementing the tool-execution loop yourself, without managing context window limits manually, and without building your own permission system from scratch.

The SDK is not an abstraction over the Anthropic Messages API. It is a complete agent runtime. When Claude decides to read a file, the SDK reads it. When Claude wants to run a shell command, the SDK runs it and returns the output. You define what tools are available and what permissions they have; the SDK handles everything else.

What You Will Build

Throughout this book, you will build progressively more sophisticated agents:

- **A read-only code analysis agent** that examines repositories for security vulnerabilities without modifying any files
- **A multi-agent code review pipeline** with specialized subagents for security scanning, style checking, and test coverage analysis

- **A RAG-powered research assistant** that searches the web, reads documents, and produces structured reports
- **A customer support agent** with human escalation paths, CRM integration, and ticket creation
- **Production deployment patterns** including sandboxed execution, session persistence, multi-tenant isolation, and observability

Each project builds on the previous ones, adding layers of sophistication while reinforcing core concepts. By the end, you will have a toolkit of patterns you can adapt to your own use cases.

How to Read This Book

This book assumes you are comfortable programming in Python (3.10 or later) or TypeScript (Node.js 18 or later). You should understand basic concepts like async/await, environment variables, and package management. No prior experience with AI agents, LLM APIs, or the Anthropic platform is required – those topics are covered from the ground up.

The book is organized into three parts:

Foundations (Chapters 1-3) establish the conceptual background and get you writing your first agent. You will learn what makes an agent different from a chatbot, how the SDK is structured, and how to use its core APIs.

Core Capabilities (Chapters 4-10) cover the building blocks of agent development: tools, memory, prompting, subagents, hooks, and structured outputs. Each chapter includes complete code examples and design patterns you can apply immediately.

Production Readiness (Chapters 11-15) addresses the challenges of deploying agents in real environments: testing, observability, security, scaling, and enterprise integration. These chapters draw on lessons from production deployments and include case studies from organizations running Claude Agent SDK-based systems at scale.

Code examples appear in both Python and TypeScript throughout the book. You can follow along in either language, though some features (particularly certain hook events) are available only in the TypeScript SDK at this time. I note these differences explicitly where they arise.

Every chapter ends with exercises to reinforce what you have learned and interview questions that test your understanding of key concepts. The references section at the end of the book provides links to official documentation, research papers, and additional resources for deeper exploration.

The one thing you need before starting is an Anthropic API key. You can obtain one from the Anthropic Console. The examples in this book use the `ANTHROPIC_API_KEY` environment variable for authentication, which is the standard approach recommended by Anthropic.

Chapter 1: AI Agent Fundamentals

The Anatomy of an AI Agent

An AI agent is a system that uses a language model as its reasoning engine to autonomously accomplish tasks in an environment. The word "autonomously" is the key distinction from a standard chatbot. A chatbot generates a response and stops. An agent generates a response, acts on it, observes the result, and decides what to do next – potentially repeating this cycle dozens or hundreds of times before producing a final output.

Every AI agent, regardless of framework or implementation, consists of four core components:

Perception. The agent must be able to observe its environment. In the context of the Claude Agent SDK, perception happens through tools. The agent reads files, searches codebases, executes commands, and fetches web pages. Each tool call is a sensory organ that provides information about the state of the world.

Reasoning. The language model serves as the reasoning engine. Given what it has perceived and what it is trying to accomplish, the model decides what action to take next. This is not a single decision but an ongoing deliberation process: "I have read three files so far; I need to check two more before I can make a recommendation."

Action. The agent must be able to modify its environment. This could mean writing files, running commands, sending API requests, or updating databases. Actions are constrained by the tools available to the agent and the permissions granted to those tools. A read-only agent can perceive but not act destructively; a full-access agent can both perceive and modify.

Memory. The agent must retain information across its reasoning cycles. Short-term memory lives in the context window – the conversation history that includes all previous prompts, tool calls, results, and model responses. Long-term memory requires persistent storage: files on disk, database records, or vector embeddings that survive beyond a single session.

These four components form a loop: perceive, reason, act, remember, repeat. The Claude Agent SDK automates this loop for you, but understanding its structure is essential for designing effective agents and troubleshooting when they behave unexpectedly.

Consider a concrete example. You ask an agent to "find all unhandled promise rejections in this codebase and add proper error handling." Here is how the four components play out:

1. **Perception:** The agent uses `Grep` to search for `.then(` patterns, `Glob` to find JavaScript files, and `Read` to examine suspicious code.
2. **Reasoning:** The model analyzes each finding, determines which promises lack error handling, and plans the fixes needed.
3. **Action:** The agent uses `Edit` to add `.catch()` handlers or convert to async/await with try/catch blocks.
4. **Memory:** Throughout the process, the agent maintains context about which files it has already examined and what changes it has made, preventing redundant work and ensuring consistency.

The agent may loop through this cycle 20 times before completing the task: reading a file, deciding what to change, making the edit, moving to the next file. Each loop iteration is called a "turn."

Tool Use and Function Calling

Tools are the bridge between an agent's reasoning and the external world. Without tools, an agent is trapped in its context window – it can only reason about information already present in the prompt. With tools, it can explore, act, and learn.

The concept of "function calling" (or tool use) was popularized by OpenAI's API in 2023, but Anthropic's implementation has some distinguishing characteristics. In the Claude API, tool use works as follows:

1. You define tools by specifying their name, description, and input schema (the arguments they accept).
2. You include these tool definitions in your API request alongside the user prompt.

3. Claude may respond with a `tool_use` block that specifies which tool to call and what arguments to pass.
4. Your application executes the tool, collects the result, and sends it back as a `tool_result` block.
5. Claude receives the result and continues reasoning – potentially calling more tools or producing a final text response.

This cycle can repeat many times within a single API call. Claude may call multiple tools in parallel (if they are independent) or sequentially (if later calls depend on earlier results). The model decides when it has enough information to stop calling tools and produce a final answer.

The Claude Agent SDK automates this entire cycle. You do not manually send `tool_result` blocks back to the API; the SDK does it for you. You define what tools are available, and the SDK handles the routing, execution, and result collection. This is fundamentally different from using the raw Messages API, where you implement the tool loop yourself.

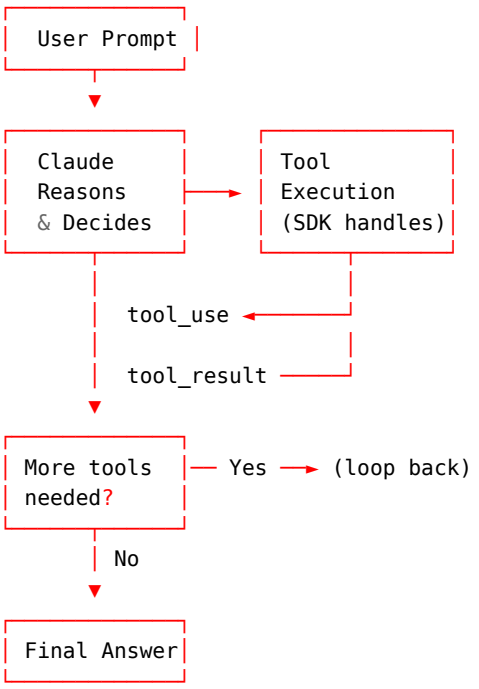
The built-in tools in the Agent SDK fall into several categories:

- **File operations:** `Read`, `Write`, `Edit` – for examining and modifying files
- **Search and discovery:** `Glob` (pattern-based file finding), `Grep` (regex search across files)
- **Command execution:** `Bash` (shell commands), `Monitor` (background processes)
- **Web access:** `WebSearch` (search the web), `WebFetch` (fetch and summarize web pages)
- **User interaction:** `AskUserQuestion` (multiple-choice questions during execution)
- **Orchestration:** `Agent` (spawn subagents), `Skill` (invoke agent skills)

Every tool consumes context window space because its definition (name, description, schema) must be included in every API call. This means that adding tools has a cost: more tools mean more tokens spent on definitions rather than conversation. The SDK addresses this at scale with "tool search," which dynamically loads only the tools relevant to the current task.

The Agent Loop

The agent loop – sometimes called the "ReAct" pattern (Reason-Act) or the "think-act-observe" cycle – is the fundamental execution model of AI agents. It describes how an agent iteratively reasons about a task, takes actions through tools, observes results, and repeats until completion.



Before the Agent SDK, developers had to implement this loop manually. A typical implementation looked like this:

```python
# Manual tool loop (what you used to have to write)
messages = [{"role": "user", "content": prompt}]
tools = [read_tool, write_tool, bash_tool]

while True:
    response = client.messages.create(
        model="claude-sonnet-4-6",
        messages=messages,
        tools=tools,
        max_tokens=4096
    )

    if not any(block.type == "tool_use" for block in response.content):
        break  # Claude is done

    for block in response.content:
        if block.type == "tool_use":
            result = execute_tool(block.name, block.input)
            messages.append({"role": "assistant", "content": [block]})
            messages.append({"role": "user", "content": [{"type":
            ↪  "tool_result", ...}]})
```

This manual approach is error-prone. You must handle parallel tool calls, manage the growing message list, implement retry logic, track token usage, and deal with edge cases like tool execution failures. The Agent SDK eliminates all of this by running the loop internally. You call `query()` once, and the SDK handles everything until the agent produces its final answer.

A single complete evaluation-execution cycle constitutes one "turn." The `max_turns` parameter caps the number of turns, preventing runaway loops in unattended execution. Each turn includes one Claude API call (which may produce multiple tool calls) and the corresponding tool executions.

Memory Architectures for Agents

Memory is perhaps the most challenging aspect of agent design. Language models have no inherent memory beyond the context window – everything must be explicitly provided in each API call. For agents, this creates a fundamental tension: the more context you provide, the better informed the agent is, but the higher the token cost and the closer you get to context window limits.

Agent memory falls into several categories:

Working memory (context window). This is the conversation history that grows with each turn. It includes the system prompt, all user messages, all

assistant responses, all tool calls, and all tool results. For Claude models, the context window ranges from 200,000 tokens (standard) to 1,000,000 tokens (extended context, available via beta). The effective working memory is smaller than the nominal limit because you must reserve space for the model's output and for compaction operations.

Project memory (CLAUDE.md). The SDK loads project-level instructions from `CLAUDE.md` or `.claude/CLAUDE.md` files. These provide persistent context about the project: build commands, coding conventions, architecture decisions, and team-specific rules. Unlike conversation history, this content is loaded at the start of each session and remains constant throughout.

File-based memory (Memory tool). The Memory tool lets Claude create, read, update, and delete files in a `/memories` directory that persists across sessions. This is how agents "remember" things between runs: they write learned information to memory files and read them back in subsequent sessions. The memory is not automatic – the agent must explicitly decide what to save.

Vector memory (RAG). For large knowledge bases, vector embeddings enable semantic search. An agent can query a vector database to find relevant documents, inject them into the context window, and reason about them. This is typically implemented through custom tools or MCP servers rather than as a built-in SDK feature.

The choice of memory architecture depends on the use case. A code review agent needs project memory (coding standards) and working memory (the current conversation). A customer support agent needs vector memory (knowledge base articles) and file-based memory (previous interactions with the same customer). A research assistant needs web access (WebSearch, WebFetch) and working memory to synthesize findings across multiple sources.

Planning and Reasoning Patterns

Not all agent tasks are the same. Some require linear, step-by-step execution. Others benefit from exploratory reasoning, where the agent considers multiple approaches before committing. Understanding different planning patterns helps you design agents that match the complexity of their tasks.

Linear execution. The simplest pattern: the agent follows a predetermined sequence of steps. "Read file A, analyze it, write report B." This works well for

well-defined tasks with clear inputs and outputs. The Agent SDK handles this naturally through the tool-use loop.

Exploratory reasoning. The agent does not know the exact steps needed upfront. It explores the environment, gathers information, and adapts its plan as it learns more. "Find all potential security vulnerabilities in this codebase" requires the agent to first understand the codebase structure, then systematically examine different areas for different types of vulnerabilities.

Hierarchical planning. Complex tasks are broken into subtasks, each handled by a specialized agent. The Claude Agent SDK supports this through subagents: the main agent delegates specific work to child agents with their own tools, context windows, and expertise. A code review pipeline might have separate subagents for security scanning, style checking, and test coverage analysis.

Chain-of-thought. The model explicitly reasons through its decision-making process before acting. Claude's "extended thinking" feature encourages this by allocating additional tokens for internal reasoning before tool use or final output. This is particularly valuable for complex decisions where the right tool call is not obvious.

The `effort` parameter in the Agent SDK controls how much reasoning depth the model applies, with levels from `"low"` to `"max"`. Higher effort levels increase token consumption and latency but improve performance on complex tasks. The choice of effort level is a trade-off between cost, speed, and quality.

Agent vs. Workflow: When to Use Each

Anthropic's research on building effective agents distinguishes between two complementary paradigms:

Agents are dynamic, model-directed systems. You give the agent a goal and a set of tools, and it decides how to accomplish the task. The execution path emerges from the model's reasoning. Agents excel at open-ended tasks where the steps are not known in advance: debugging an unfamiliar codebase, researching a new topic, or analyzing an unstructured dataset.

Workflows are deterministic, code-directed pipelines. You write the control flow as explicit Python or TypeScript code, and each step delegates work to a fresh agent invocation. The orchestration is predetermined; only the work

inside each `query()` call is model-powered. Workflows excel at structured processes with known steps: "extract data from these files, transform it according to these rules, load it into this database."

The Claude Agent SDK supports both patterns. Single `query()` calls create agents that reason dynamically. Multi-step Python/TypeScript code that chains multiple `query()` calls creates workflows. The most effective production systems often combine both: a workflow framework with agent-powered steps, or an agent that invokes subagents to handle specific subtasks within its dynamic reasoning loop.

The key insight is that neither paradigm is universally superior. Agents are more flexible but less predictable. Workflows are more reliable but less adaptable. The best systems use the right tool for each part of the problem.

Chapter Summary

This chapter established the foundational concepts of AI agent architecture:

- **Agents** perceive through tools, reason with language models, act on their environment, and remember across sessions
- **Tools** are the bridge between reasoning and action, with the SDK automating the tool-execution loop
- **The agent loop** iterates through think-act-observe cycles until a task is complete
- **Memory** exists at multiple levels: working memory (context window), project memory (CLAUDE.md), file-based memory (Memory tool), and vector memory (RAG)
- **Planning patterns** range from linear execution to hierarchical multi-agent systems
- **Agents vs. workflows** represent a spectrum from dynamic model-directed execution to deterministic code-directed pipelines

Understanding these concepts is essential before diving into the SDK's specific APIs and features. The next chapter explores the SDK's architecture, installation, and configuration in detail.

Exercises

1. Design an agent for your own domain. What tools would it need? What memory systems are appropriate? Is it better implemented as an agent or a workflow?
2. Trace through the agent loop manually for a simple task like "find all TODO comments in this repository." List each turn, the tool calls made, and the reasoning that leads to the next action.
3. Compare the context window requirements for two agents: one that reviews a single file, and one that analyzes an entire microservice architecture. How does memory strategy differ between them?

Interview Questions

1. What are the four core components of an AI agent, and how do they interact in the agent loop?
2. Why is the Agent SDK's approach to tool execution different from the raw Messages API, and what advantages does it provide?
3. When would you choose an agent pattern over a workflow pattern, and vice versa?
4. Describe the trade-offs between working memory (context window) and file-based memory for a long-running agent.

Chapter 2: SDK Architecture and Environment Setup

The SDK Stack: Client SDK vs. Agent SDK

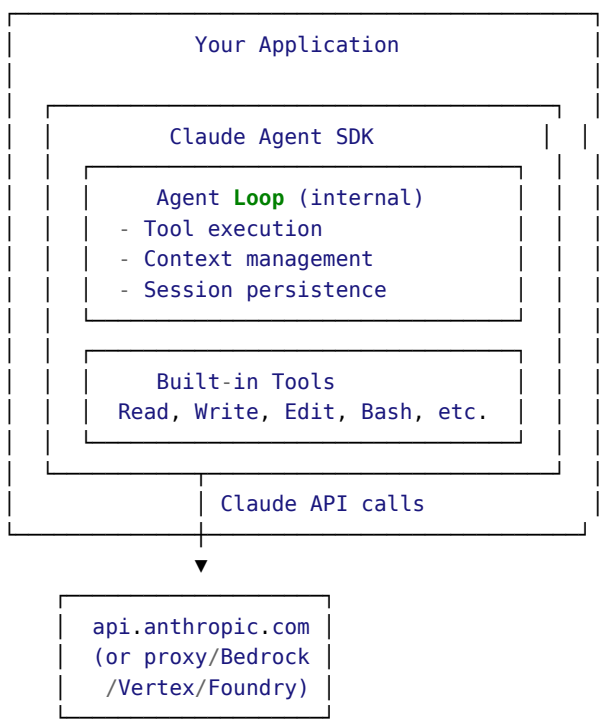
Before diving into the Agent SDK itself, it is critical to understand where it sits in Anthropic's tooling ecosystem. There are three distinct products, and confusing them is a common source of frustration for newcomers.

The Client SDK (anthropic Python package / @anthropic-ai/sdk TypeScript package) is a thin wrapper around the Messages API. You send prompts and receive responses. If you want tool use, you implement the tool-execution loop yourself: handle `tool_use` blocks, execute the requested tools, format `tool_result` blocks, and send them back in subsequent API calls. This gives you maximum flexibility but requires significant boilerplate.

The Agent SDK (claude-agent-sdk Python package / @anthropic-ai/claude-agent-sdk TypeScript package) is a complete agent runtime. It includes the tool-execution loop, context management, session persistence, permission systems, and lifecycle hooks. You call `query()` with a prompt and configuration, and the SDK runs the entire agent loop until completion, streaming typed messages back to your application.

Managed Agents are Anthropic-hosted agent sessions accessible via a REST API. They run on Anthropic's infrastructure with managed sandboxes, event logs, and asynchronous session handling. You do not manage the runtime; you send requests and receive results. Managed Agents are the production destination for many SDK prototypes, but they require a different integration pattern.

The Agent SDK sits in the middle: it gives you the full power of Claude Code's agent harness without requiring you to host it on Anthropic's infrastructure. You run it in your own process, on your own machines, with access to your local filesystem and services.



Installation and Authentication

The Agent SDK is available for both Python and TypeScript. Choose the language that best fits your application's existing stack – the features are largely equivalent, though some hook events are TypeScript-only at this time.

Python installation:

```
pip install claude-agent-sdk
```

The Python package requires Python 3.10 or later. It bundles a native Claude Code binary pinned to the SDK's package version, so you do not need to install the CLI separately.

TypeScript installation:

```
npm install @anthropic-ai/claude-agent-sdk
```

The TypeScript package requires Node.js 18 or later. It also optionally bundles a native Claude Code binary as a dependency.

Authentication uses environment variables. The standard approach is to set `ANTHROPIC_API_KEY` with your API key from the Anthropic Console:

```
export ANTHROPIC_API_KEY="sk-ant-api03-..."
```

The SDK reads this variable automatically. You never pass the API key directly in code – that would be a security risk.

Third-party providers are supported through environment variables that route API calls without changing your agent code:

Provider	Environment Variable	Additional Configuration
Amazon Bedrock	`CLAUDE_CODE_USE_BEDROCK=1`	AWS credentials via standard mechanisms
Claude Platform on AWS	`CLAUDE_CODE_USE_ANTHROPIC_AWS=1`	Plus `ANTHROPIC_AWS_WORKSPACE_ID`
Google Cloud Vertex AI	`CLAUDE_CODE_USE_VERTEX=1`	GCP credentials via standard mechanisms
Microsoft Azure Foundry	`CLAUDE_CODE_USE_FOUNDRY=1`	Azure credentials via standard mechanisms

When using third-party providers, you must match model IDs to the provider's catalog. For example, Bedrock uses model identifiers like `anthropic.claude-sonnet-4-6` rather than the Anthropic API's `claude-sonnet-4-6`. Do not mix conflicting credentials – set only one provider variable at a time.

Anthropic explicitly prohibits third-party developers from leveraging claude.ai login credentials or subscription rate limits for products built

with the Agent SDK. API key authentication is mandatory for any shipped application.

Project Configuration and CLAUDE.md

The Agent SDK loads configuration from the filesystem by default, mirroring Claude Code's behavior. This includes project-level instructions, skills, commands, and memory files. Understanding this configuration system is essential for reproducible agent behavior.

The `.claude/` directory structure:

```
your-project/
├── .claude/
│   ├── CLAUDE.md          # Project-level instructions
│   ├── skills/
│   │   └── my-skill/
│   │       └── SKILL.md   # Skill definition
│   └── commands/
│       └── review.md      # Legacy slash command (deprecated)
├── src/
├── tests/
└── package.json
```

CLAUDE.md is the most important configuration file. It provides persistent project context that loads at the start of each session. Typical contents include:

```
# Project Context

# Build Commands
- `npm install` to install dependencies
- `npm test` to run the test suite
- `npm run build` to compile TypeScript

# Coding Conventions
- Use TypeScript strict mode
- Prefer async/await over promises
- All public functions must have JSDoc comments
- Maximum line length: 100 characters

# Architecture
- This is a REST API built with Express
- Database: PostgreSQL 16
- Authentication: JWT tokens via auth middleware
- Error handling: centralized error middleware in src/middleware/errors.ts
```

The SDK loads CLAUDE.md when `setting_sources` includes `"project"` (the default). The content is injected into the conversation – not the system prompt – ensuring compatibility with any prompt configuration.

Controlling filesystem loading: The `setting_sources` option (Python) or `settingSources` option (TypeScript) controls which configuration files the SDK loads:

- `["user", "project", "local"]` (default): Load all filesystem configurations
- `["project"]`: Load only project-level configs (CLAUDE.md, skills in `.claude/skills/`)
- `[]`: Disable all filesystem configurations – essential for CI/CD and multi-tenant isolation

Disabling filesystem loading is a common source of reproducibility issues in continuous integration. If your agent behaves differently in CI than locally, check whether `setting_sources` is inadvertently loading user-level configurations.

The Subprocess Model

The Agent SDK deploys as a long-lived process rather than a stateless API wrapper. Understanding the subprocess model is critical for production deployment:

- Each `query()` call spawns a separate `claude` CLI subprocess
- The subprocess communicates with Claude via the Messages API and manages its own shell, working directory, and JSONL session transcripts
- One agent session equals one subprocess; concurrent sessions require separate processes
- You must pass the `cwd` option to isolate filesystems between sessions

This model has important implications:

Concurrency. If you need to run multiple agents simultaneously, each gets its own subprocess. This means memory usage scales linearly with concurrency. A host with 16 GiB of RAM might comfortably run 10-15 concurrent agent sessions (assuming ~1 GiB per session baseline).

State persistence. Session transcripts, CLAUDE.md memory files, and working directory artifacts persist on local disk until the container restarts, scales down, or migrates nodes. For production durability, you need a `SessionStore` adapter to mirror transcripts to external storage.

Isolation. Each subprocess has its own working directory (set via `cwd`). If two sessions share the same working directory without proper isolation, they can interfere with each other's file operations and session state. Always use unique `cwd` values for multi-tenant deployments.

Multi-Language Support: Python and TypeScript

The Agent SDK offers feature parity between Python and TypeScript for most capabilities, but there are important differences to be aware of:

Naming conventions. The Python SDK uses camelCase for multi-word option fields that match the wire format, rather than following Python's typical snake_case convention. For example, `allowed_tools`, `disallowed_tools`, `setting_sources`, and `max_turns` use underscores in Python but correspond to camelCase fields (`allowedTools`, `disallowedTools`, `settingSources`, `maxTurns`) in the TypeScript SDK. This is intentional – it prevents serialization mismatches.

Hook events. The TypeScript SDK supports additional observability events beyond what the Python SDK currently provides. Specifically, `SessionStart` and `SessionEnd` hooks are available in TypeScript but not as Python SDK callbacks (they require shell command configuration via `setting_sources` in Python).

Stateless execution. TypeScript supports `persistSession: false` to keep sessions in memory only. Python always persists sessions to disk, which may be undesirable in some serverless environments.

Structured output helpers. TypeScript developers can use Zod schemas for automatic type-safe output validation, while Python developers use Pydantic models. Both approaches generate JSON Schema automatically and provide compile-time/type-checking safety.

When to choose each language:

- Choose **Python** if your team's primary stack is Python, if you are integrating with data science or ML tooling, or if you prefer the `ClaudeSDKClient`

pattern for multi-turn stateful interactions.
- Choose **TypeScript** if your application is Node.js-based, if you need the additional hook events, or if you want stateless session options for serverless deployment.

Chapter Summary

This chapter covered the SDK's architecture and environment setup:

- The Agent SDK sits between the Client SDK (manual tool loops) and Managed Agents (hosted runtime), providing a complete agent runtime in your own process
- Installation requires Python 3.10+ or Node.js 18+, with authentication via `ANTHROPIC_API_KEY`
- Third-party providers (Bedrock, Vertex, Foundry) are supported through environment variables
- Project configuration loads from `.claude/` directories, with CLAUDE.md providing persistent project context
- The subprocess model means each `query()` spawns a separate process, with implications for concurrency and state management
- Python and TypeScript offer largely equivalent features, with some differences in naming conventions, hook events, and session persistence

With the environment set up and the architecture understood, the next chapter dives into the core APIs: `query()`, `ClaudeSDKClient`, message types, turn management, and session handling.

Exercises

1. Install the Agent SDK in both Python and TypeScript. Write a minimal "hello world" agent that prints each message type it receives.
2. Create a `.claude/CLAUDE.md` file for a sample project. Include build commands, coding conventions, and architecture notes. Verify that the SDK loads it by running an agent with `setting_sources=["project"]`.
3. Experiment with `setting_sources=[]` to disable all filesystem loading. Observe how agent behavior changes without CLAUDE.md context.

Interview Questions

1. What is the difference between the Client SDK and the Agent SDK, and when would you use each?
2. Why does each `query()` call spawn a separate subprocess, and what are the implications for production deployment?
3. How do you control which filesystem configurations the SDK loads, and why might you want to disable them entirely?
4. What authentication options does the Agent SDK support, and why is claude.ai login prohibited for third-party products?

Chapter 3: Core APIs and the Agent Loop

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

The query() Function

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

ClaudeSDKClient for Multi-Turn Conversations

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Message Types and Lifecycle

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Turn Management and Budget Controls

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Session Persistence and Resumption

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

The Agent Loop in Action: First Complete Example

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Chapter Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Exercises

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Interview Questions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Chapter 4: Built-In Tools and Tool Use

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

File Operations: Read, Write, Edit

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Search and Discovery: Glob, Grep

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Command Execution: Bash and Monitor

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Web Tools: WebSearch and WebFetch

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

User Interaction: AskUserQuestion

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Permission Modes and Tool Access Control

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Parallel vs. Sequential Tool Execution

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Chapter Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Exercises

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Interview Questions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Chapter 5: Custom Tools and MCP Integration

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Defining Custom Tools in-Process

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Tool Result Formats

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

MCP Server Integration

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Tool Search and Context Optimization

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Building a Complete Custom Tool: Weather Service Example

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Chapter Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Exercises

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Interview Questions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Chapter 6: Memory Systems and Context Management

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

The Context Window: Sizes, Limits, and Budgets

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Automatic Context Compaction

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

CLAUDE.md: Project-Level Memory

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

The Memory Tool

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Retrieval-Augmented Generation with Agents

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Context Optimization Strategies

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Chapter Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Exercises

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Interview Questions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Chapter 7: Prompt Engineering for Agents

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

System Prompts vs. User Prompts in Agent Context

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Writing Effective Agent System Prompts

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

CLAUDE.md Mastery

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

XML Structuring and Few-Shot Examples

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Extended Thinking and Effort Levels

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Prompt Caching Across Sessions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Chapter Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Exercises

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Interview Questions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Chapter 8: Subagents and Multi-Agent Systems

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Why Subagents: Context Isolation and Parallelism

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Programmatic AgentDefinition

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Automatic vs. Explicit Invocation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Subagent Inheritance and Isolation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Detecting and Resuming Subagents

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Dynamic Workflows for Large-Scale Orchestration

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Common Multi-Agent Patterns

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Chapter Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Exercises

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Interview Questions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Chapter 9: Hooks and Human-in-the-Loop Patterns

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Hook Architecture and Execution Pipeline

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Lifecycle Events Reference

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Matcher Patterns and Filtering

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Blocking and Transforming Tool Calls

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Human Approval Workflows

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Async Hooks for Observability

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Anti-Patterns: Hook Pitfalls

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Chapter Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Exercises

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Interview Questions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Chapter 10: Structured Outputs and Data Extraction

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Why Structured Outputs Matter

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Defining Output Schemas

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Multi-Tool Structured Extraction

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Error Handling for Schema Validation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Best Practices and Anti-Patterns

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Chapter Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Exercises

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Interview Questions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Chapter 11: Testing, Debugging, and Evaluation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

The Challenge of Testing Agents

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Unit Testing Tools and Hooks

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Promptfoo Integration

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

MLflow Tracing and Evaluation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Debugging Strategies

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Building Evaluation Datasets

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Continuous Evaluation in CI/CD

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Chapter Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Exercises

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Interview Questions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Chapter 12: Observability and Performance Optimization

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

The Three Signals: Metrics, Traces, Logs

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

OpenTelemetry Integration

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Cost Tracking and Budget Management

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Performance Optimization Strategies

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Prompt Caching for Cost Reduction

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Observability Tooling Comparison

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Redacting PII and Secrets from Telemetry

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Chapter Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Exercises

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Interview Questions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Chapter 13: Security and Sandboxing

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

The Agent Threat Model

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Built-In Security Features

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Sandbox Runtime: bubblewrap and sandbox-exec

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Container Isolation Strategies

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Credential Management

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Filesystem Security

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Network Security

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Defense in Depth Checklist

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Chapter Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Exercises

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Interview Questions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Chapter 14: Deployment, Scaling, and Production Operations

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Hosting Patterns

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Container Providers

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Session Persistence in Production

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Scaling Strategies

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Multi-Tenant Isolation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Monitoring and Alerting in Production

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Known Limitations and Workarounds

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Chapter Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Exercises

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Interview Questions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Chapter 15: Real-World Case Studies and Enterprise Architecture

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Case Study: Automated Code Review Pipeline

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Case Study: RAG-Powered Customer Support Agent

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Case Study: Research Assistant with Parallel Subagents

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Enterprise Integration Patterns

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Managed Agents vs. Self-Hosted SDK

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Comparison with Alternative Frameworks

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

The Future of Agent Development

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Chapter Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Exercises

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Interview Questions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Conclusion: Your Agent Journey Ahead

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Key Takeaways

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Building Your First Production Agent

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

The Evolving Landscape

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

Resources and Community

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.

References

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/buildingaiagentswiththeclaudeagentsdk.