

Claude Code Handbook for Software Engineers

An Engineering Reference for
Agentic Development



Architecture



Workflows



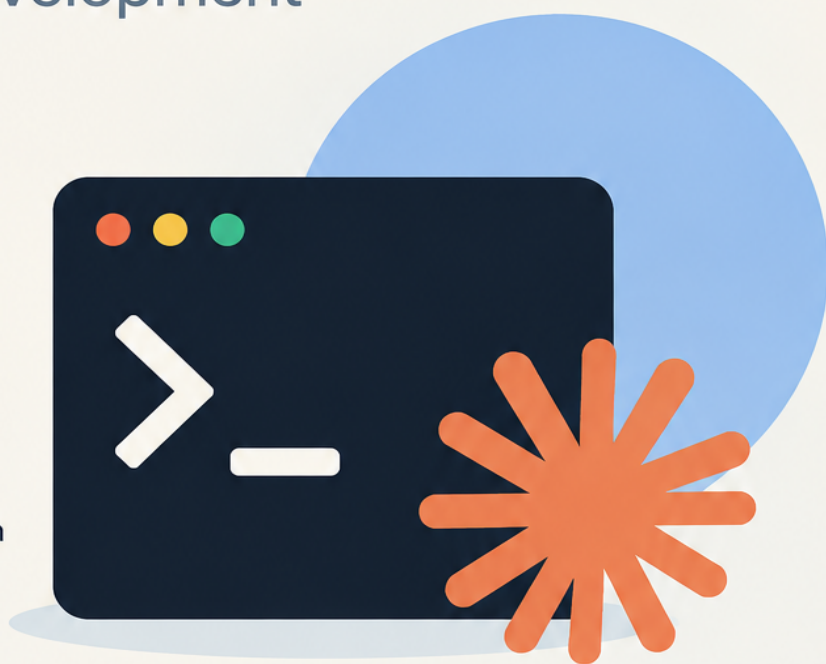
Security



Team Adoption



Advanced
Patterns



Emil Novak

Claude Code Handbook for Software Engineers

An Engineering Reference for Agentic Development

Steve Publications

This book is available at

<https://leanpub.com/claudecodehandbookforsoftwareengineers>

This version was published on 2026-09-28



This is a [Leanpub](#) book. Leanpub empowers authors and publishers with the Lean Publishing process. [Lean Publishing](#) is the act of publishing an in-progress ebook using lightweight tools and many iterations to get reader feedback, pivot until you have the right book and build traction once you do.

© 2026 Steve Publications

Contents

- An Engineering Reference for Agentic Development 1**
- Chapter 1: What Is Claude Code 2**
 - Beyond Chat: From Copilot to Agentic Coding 2
 - The Claude Code Architecture 3
 - Models, Tokens, and Capability Boundaries 4
 - The Developer-Agent Relationship 6
 - What Claude Code Is Not 7
- Chapter 2: Installation and Environment Setup 9**
 - Prerequisites and System Requirements 9
 - Installation Methods: Native, Package Managers, npm 10
 - Authentication: OAuth, API Keys, and Cloud Providers 12
 - Verifying Installation and Diagnostics 13
 - Environment Considerations: Containers, WSL, Remote Systems . . . 15
- Chapter 3: CLI Usage and Core Commands 18**
 - Interactive Sessions and Terminal Integration 18
 - Non-Interactive Mode: One-Shot Execution and Scripting 18
 - Essential Flags: Model, Effort, Permission Mode, and Bare Mode . . . 18
 - System Prompts and Customization 18
 - Continuation, History, and Session Management 18
- Chapter 4: Configuration and Settings 19**
 - Settings Files and Precedence 19
 - Permissions: Allow, Ask, Deny Rules 19
 - Environment Variables and Process Configuration 19
 - Model and Effort Configuration 19
 - Debugging Configuration and Troubleshooting 19
- Chapter 5: CLAUDE.md and Project Instructions 20**

CONTENTS

CLAUDE.md Fundamentals and Scope	20
Writing Effective Project Instructions	20
Modular Rules and the .claude/rules Directory	20
Auto Memory and Persistent Context	20
Instructions Hierarchy and Conflict Resolution	20
Chapter 6: Tools and Capabilities	21
The Tool Architecture and Permission Model	21
Filesystem Tools: Read, Write, Edit, Glob, Grep	21
Shell Tools: Bash, PowerShell, and Command Execution	21
Web Tools: Search and Fetch	21
Agent Tools and Specialized Capabilities	21
Chapter 7: Agents and Subagents	22
Agent Spawning and the Task Tool	22
Subagent Configuration and Isolation	22
Parallel and Sequential Workflows	22
Dynamic Workflows at Scale	22
When Multi-Agent Overhead Is Worth It	22
Chapter 8: MCP and External Integrations	23
Understanding Model Context Protocol	23
Adding and Managing MCP Servers	23
Common Integrations: GitHub, Databases, APIs	23
Security and Credential Management for MCP	23
Build Your Own MCP Server	23
Chapter 9: Skills, Plugins, and Custom Commands	24
Skills: Structure and Invocation	24
Custom Slash Commands	24
Plugins and the Marketplace Ecosystem	24
Skill Evaluation and Testing	24
Chapter 10: Context Engineering	25
Context Windows and Token Economics	25
Information Prioritization Strategies	25
Repository Discovery and Codebase Indexing	25
Persistent vs. Transient Context	25
Stale Context and Its Dangers	25

- Chapter 11: Prompt Design and Task Specification 26**
 - Writing Specifications Claude Code Can Execute 26
 - Task Decomposition and Batching 26
 - Constraints and Guardrails 26
 - Verification Criteria and Acceptance Tests 26
 - Avoiding Ambiguity and Scope Creep 26
- Chapter 12: Implementation Workflows 27**
 - Feature Implementation from Requirements 27
 - Iterative Development and Feedback Loops 27
 - Refactoring with Confidence 27
 - Debugging and Incident Investigation 27
 - Handling Partial or Incorrect Implementations 27
- Chapter 13: Testing and Quality Assurance 28**
 - Test-Driven Development with Claude Code 28
 - Generating Tests: Unit, Integration, and E2E 28
 - Test Quality and Coverage 28
 - Static Analysis and Linting 28
 - Performance and Security Testing 28
- Chapter 14: Git Workflows and Code Review 29**
 - Branch Strategies and Claude Code 29
 - Commits and Conventional Messages 29
 - Pull Request Workflows 29
 - AI-Assisted Code Review 29
 - Maintaining Review Quality 29
- Chapter 15: CI/CD and Automation 30**
 - Claude Code GitHub Actions Setup 30
 - Automated PR Review and Response 30
 - CI Integration Patterns 30
 - Automation Safety and Rate Limits 30
 - Self-Hosted Runners and On-Premise Options 30
- Chapter 16: Security and Governance 31**
 - Permission Model and Least Privilege 31
 - Sandboxing and Execution Isolation 31
 - Secrets, Credentials, and API Keys 31
 - Prompt Injection and Adversarial Attacks 31

Supply Chain and Dependency Security	31
Data Privacy and Compliance	31
Security Summary Checklist	32
Chapter 17: Enterprise and Team Adoption	33
Managed Settings and Policy Enforcement	33
Team Conventions and Shared Instructions	33
Onboarding and Training	33
Collaboration Patterns	33
Measuring Adoption and Effectiveness	33
Chapter 18: Advanced Engineering Patterns	34
Working in Monorepos	34
Polyglot and Mixed-Technology Codebases	34
Legacy Code Modernization	34
Large-Scale Architectural Refactoring	34
Observability and Production Systems	34
Chapter 19: Case Studies and Workflows	35
Building a REST API Service from Scratch	35
Adding Authentication to an Existing Application	35
Migrating a Legacy Module to Modern Architecture	35
Investigating and Fixing a Production Incident	35
Chapter 20: Trade-offs, Limitations, and the Path Forward	36
Where Claude Code Excels	36
Where It Is Unreliable	36
Tasks That Demand Human Judgment	36
Cost, Complexity, and Long-Term Maintenance	36
The Future of Agentic Development	36
Honest Assessment	36
Conclusion: Engineering with Intelligence	38
References	39

An Engineering Reference for Agentic Development

This book is a comprehensive, technically authoritative guide to Claude Code for professional software engineers. It covers the full spectrum from fundamental concepts through production-grade implementation, including architecture, workflows, security, team adoption, and advanced patterns. Every chapter provides concrete commands, working code examples, and engineering reasoning designed for engineers who want to integrate Claude Code into real-world development. The material is written assuming familiarity with programming, Git, APIs, and standard software development practices, and progresses rapidly into advanced territory. The goal is not to sell you on a tool but to give you the knowledge to use it effectively, reliably, and safely.

Chapter 1: What Is Claude Code

Beyond Chat: From Copilot to Agentic Coding

You have probably used GitHub Copilot or another inline AI assistant. You type some code, and the IDE suggests the next few lines. You accept or reject the suggestion. The flow is simple and contained: you are always the one typing, the one in control, and the assistant is offering hints.

Claude Code is fundamentally different. It does not sit beside your editor offering suggestions. It sits in your terminal and acts. You tell it what you want done. It reads files, modifies files, runs commands, searches the codebase, commits changes, opens pull requests, and runs tests. It can do all of this without you ever touching the keyboard again, executing a sequence of actions that extends over minutes or hours instead of seconds. [1,2]

This shift from suggestion to execution changes the relationship with the tool. With Copilot you are still the primary implementer. With Claude Code, the tool is the implementer and you are the specifier, reviewer, and decision-maker. This is what “agentic coding” means: the AI does not merely generate text that looks like code. It perceives its environment through tools, plans actions, executes those actions, and observes the results. It can recover from errors, adjust its approach, and continue working toward a goal without constant human intervention.

Consider a concrete scenario. You want to add rate limiting to an Express API. With a chat-based tool you would ask for code, paste it into a file, fix any issues, run the server, and repeat. With Claude Code you would describe the requirement: add rate limiting to the API endpoints using Redis, one thousand requests per minute per IP, with appropriate error responses. Claude Code would then:

- Read your existing codebase to understand the structure
- Install the necessary dependencies
- Create a rate limiting middleware
- Wire it into your Express app

- Run the server and verify it works
- Optionally write tests for the new behavior

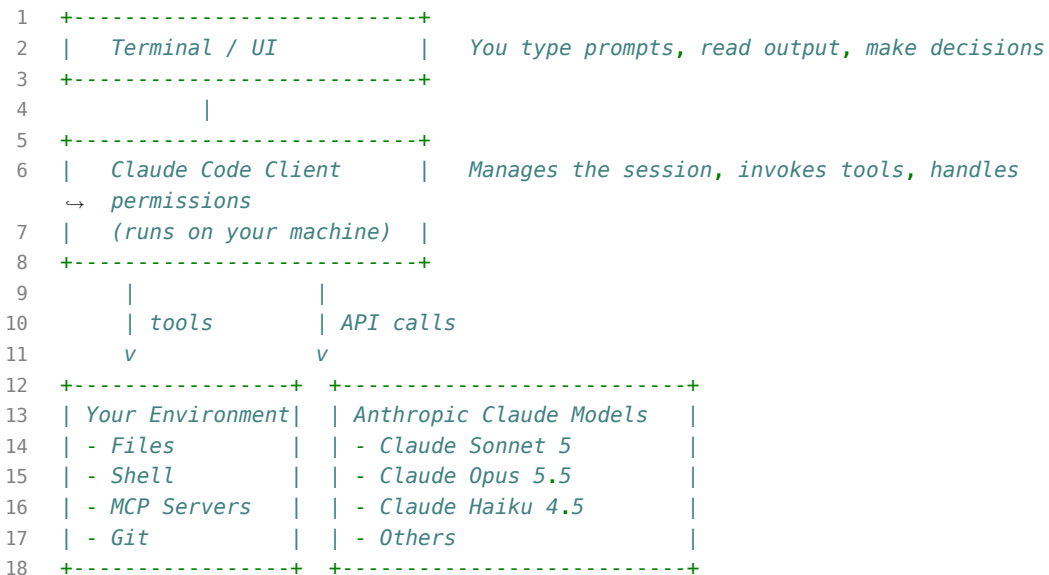
You review the changes. You approve or reject them. But the tool has done the work, not you.

This is not magic. It is a specific architecture with specific capabilities and specific failure modes. Understanding those is what this book is about.

The Claude Code Architecture

Claude Code is a client application that runs locally on your machine. It connects to Anthropic Claude models over the API, and it provides those models with access to tools that interact with your local environment. This is an important distinction: Claude Code itself is not the intelligence. It is the harness that gives a language model the ability to perceive and act. [3,4]

The architecture has several layers:



When you launch Claude Code and type a prompt, the following sequence occurs:

1. The client assembles a system prompt containing configuration, project instructions, and tool definitions

2. Your prompt is appended to the conversation history
3. The combined request is sent to a Claude model
4. The model generates a response, which may include tool calls
5. If the response includes tool calls, the client executes those tools locally and returns the results to the model
6. The model generates another response based on the tool results
7. This loop continues until the model produces a final response without tool calls

At each step, Claude Code applies permission checks. It may ask you for approval before executing certain actions, depending on how it is configured.

This architecture has several implications:

- Claude Code needs your permissions to modify your files and run commands on your machine. It is not sandboxed by default.
- The model sees tool output verbatim. If a tool returns sensitive data, the model sees it.
- The model is stateless between API calls. Claude Code maintains state by including conversation history and context in each request.
- The quality of Claude Code depends on the underlying model and the quality of the information available to it.

Models, Tokens, and Capability Boundaries

Claude Code uses Claude models. The model you choose affects capabilities, latency, and cost. As of mid 2026, Anthropic offers several models for Claude Code:

- **Claude Opus 5.5** and **Claude Opus 5**: The most capable reasoning models, designed for complex tasks. They perform well on large-scale refactoring, architectural decisions, and multi-file changes. They are slower and more expensive than other models. Opus 5.5 leads in terminal benchmarking scores and supports fast mode for higher output speed at premium pricing. [21]

- **Claude Sonnet 5:** The default choice for most Claude Code sessions. Sonnet offers strong reasoning capabilities at lower cost and latency than Opus. It handles implementation tasks, debugging, and multi-file changes reliably. For many engineering workflows, Sonnet provides the best balance of capability, speed, and cost. [21]
- **Claude Haiku 4.5:** A fast, cheap model for simpler tasks like code review, documentation, and single-file edits. It is not suited for complex reasoning or large-scale changes.
- **Fable 5.1:** A high-end reasoning model designed for particularly complex tasks. It is expensive but performs exceptionally well on difficult problems.

Each model has a maximum context window. Claude Opus 5 and Sonnet 5 support up to 200,000 input tokens and can produce up to 100,000 output tokens in a single response when using the appropriate output configuration. Claude Haiku 4.5 has a smaller window. The context window limits how much information the model can process at once. In Claude Code, the context window is consumed by:

- The system prompt (instructions, tool definitions, configuration)
- Project instructions (CLAUDE.md, rules, memory)
- Conversation history
- File contents read during the session
- Tool output

When the context window fills, Claude Code performs compaction: it summarizes earlier parts of the conversation to free space. This means that details from the beginning of a long session may become less precise over time. Understanding this is critical: if you give Claude Code a complex requirement at the start of a session and then spend an hour fixing edge cases, the model may lose some fidelity in how it remembers the original requirement. You can recover by restating it or ensuring it is captured in a persistent instruction file.

The concept of “effort” is also important. Claude Code supports different effort levels:

- **Low:** Minimal reasoning. Fast and cheap. Suitable for simple lookups or trivial changes.
- **Medium:** Standard reasoning. The default for most tasks.

- **High:** Deep reasoning. Useful for complex problems, architectural decisions, or multi-file refactoring.
- **Ultracode:** Maximum effort. Available on the newest models, this level may trigger dynamic workflows with multiple subagents operating in parallel. It is expensive but can handle very large tasks.

Choosing the right model and effort level is part of effective Claude Code usage. Using Opus at ultracode effort for a single-file bug fix wastes money. Using Haiku for a complex architectural migration will produce poor results.

The Developer-Agent Relationship

The most important conceptual shift with Claude Code is understanding your role relative to the agent. You are no longer the sole implementer. You are:

- **The specifier:** You define what needs to be done. The quality of the specification determines the quality of the result.
- **The reviewer:** You review every change the agent makes. You are responsible for accepting or rejecting those changes.
- **The architect:** You define the overall architecture, constraints, and patterns. The agent works within those boundaries.
- **The final decision-maker:** The agent proposes, you dispose. You always retain authority over what lands in the codebase.

This relationship requires a different mindset than working with Copilot. With Copilot you might accept a suggestion without fully reading it because you trust your own ability to catch mistakes. With Claude Code, the agent is doing substantially more work autonomously. The stakes are higher. You cannot review code you never see, so Claude Code is designed to show you changes before they are applied. But it is still your responsibility to look.

The best way to think about Claude Code is not as a replacement for you, but as a powerful junior engineer with an unusual set of capabilities. It can read your entire codebase faster than any human. It can search across thousands of files instantly. It can generate boilerplate and glue code at high speed. It can work without sleep or breaks. But it can also hallucinate, miss subtle requirements, apply patterns where they do not belong, and make changes that are locally correct but globally harmful.

Your job is to harness its strengths while managing its weaknesses. You do that through clear specifications, explicit constraints, verification checks, and disciplined review. The rest of this book is about how to do that.

What Claude Code Is Not

It is important to clarify what Claude Code is not, because misunderstandings here lead to problems.

Claude Code is not a replacement for engineering judgment. It has no understanding of business context, stakeholder priorities, or strategic trade-offs. It optimizes for what you ask it to do, not for what should be done. If you ask it to optimize performance without constraints, it may produce code that is fast but unreadable or fragile. It cannot tell you whether to refactor or rewrite, whether to add a feature or cut one, or whether a technical decision aligns with business goals. Those decisions require human judgment.

Claude Code is not a testing framework. It can write tests, run tests, and help with debugging, but it does not replace a proper testing strategy. Tests it writes may miss edge cases, may duplicate existing coverage, or may assert the wrong things. They are suggestions, not guarantees.

Claude Code is not an architecture tool. It can propose designs, but it does not have a holistic understanding of your organization's constraints, non-functional requirements, or historical context. It can suggest patterns it has seen in training data, but whether those patterns fit your situation is your decision.

Claude Code is not secure by default. It runs with your permissions. It can read your files, run your commands, and access your credentials if they are in scope. It can be tricked by prompt injection attacks. It does not have an intrinsic security model; you must build one around it.

Claude Code is not deterministic. Running the same prompt twice may produce different results. The results may be equally good, one may be better, or one may be wrong. This is an inherent property of probabilistic models. You cannot rely on exact reproducibility. You can improve consistency through better specifications and constraints, but you cannot eliminate the non-determinism.

Claude Code is not always right. Even when it produces code that compiles, runs, and passes tests, it may be wrong in ways you do not immediately

detect. It may introduce subtle performance regressions, memory leaks, race conditions, or security vulnerabilities. You must treat its output as if it were written by another developer on your team: review it, test it, and hold it to the same standards.

Understanding these limitations is not pessimism. It is the foundation of responsible usage. Claude Code is a powerful tool when used correctly. Using it incorrectly can cause real damage. The next chapters will show you how to use it correctly.

Chapter 2: Installation and Environment Setup

Prerequisites and System Requirements

Claude Code runs as a native application on your machine. It requires a terminal to operate, either your system terminal or an integrated terminal within your IDE. Before installing, verify your system meets the requirements.

Supported Operating Systems:

- **macOS 13.0 or later** (Ventura and newer) on x64 or ARM64
- **Windows 10 version 1809 or later**, Windows 11, and Windows Server 2019 or later
- **Linux:** Ubuntu 20.04+, Debian 10+, Alpine 3.19+, and most modern distributions [2]

Hardware Requirements:

- At least 4 GB of RAM. Claude Code itself is lightweight, but running it alongside your development environment and IDE means you should have headroom.
- Internet connectivity. Claude Code requires network access to call the Claude API.
- Sufficient disk space for the binary and its data directory.

Alpine Linux Considerations:

If you are running Alpine Linux, there are additional requirements. Alpine uses musl libc instead of glibc, so you need to install bash, curl, libgcc, and libstdc++ explicitly. You may also need to set `USE_BUILTIN_RIPGREP=0` in your settings if the bundled ripgrep has issues.

```
1 # Alpine Linux prerequisites
2 apk add bash curl libgcc libstdc++ ripgrep
```

Network Configuration:

Claude Code needs outbound HTTPS access to Anthropic API endpoints. If you are behind a corporate proxy, configure environment variables:

```
1 export HTTPS_PROXY=https://your-proxy.example.com:8080
2 export NO_PROXY=localhost,127.0.0.1
```

Installation Methods: Native, Package Managers, npm

Anthropic recommends the native installer as the primary installation method. It is self-updating, signed, and works consistently across platforms. There are several valid installation methods. Choose one and stick with it.

Native Installer (Recommended)

The native installer downloads and installs Claude Code automatically, placing it in your system path. It also sets up auto-updating.

On macOS and Linux:

```
1 curl -fsSL https://claude.ai/install.sh | bash
```

On Windows PowerShell:

```
1 irm https://claude.ai/install.ps1 | iex
```

On Windows CMD:

```
1 curl -fsSL https://claude.ai/install.cmd -o install.cmd && install.cmd && del
   ↪ install.cmd
```

After installation, the `claude` command should be available in your terminal.

Homebrew (macOS and Linux)

If you prefer Homebrew, you can install Claude Code as a cask:

```
1 brew install --cask claude-code
```

This installs a stable version. To get the latest version:

```
1 brew install --cask claude-code@latest
```

WinGet (Windows)

On Windows, use WinGet:

```
1 winget install Anthropic.ClaudeCode
```

npm (Deprecated)

Anthropic previously distributed Claude Code as an npm package (@anthropic-ai/claude-code). This method is now deprecated. The npm package exists but may not receive updates or may be removed. Do not use npm for new installations.

Linux Package Managers

Claude Code is available through package repositories for apt, dnf, and apk. These repositories provide both stable and latest channels.

Desktop Application

Claude Code also offers a desktop application for macOS, Windows, and Linux. The desktop app wraps the CLI in a graphical interface and provides additional features like dedicated workspaces and artifact publishing. It is a valid alternative to the terminal-based workflow, particularly for teams transitioning from IDE-based AI tools.

Verification

After installation, verify that Claude Code is working:

```
1 claude --version
```

This should display the version number. For more detailed diagnostics:

```
1  claude doctor
```

This command checks your installation, authentication status, and configuration for issues.

Binary Integrity

Claude Code binaries are signed. Anthropic publishes its GPG key fingerprint: 31DDDE24DDFAB679F42D7BD2BAA929FF1A7ECACE. On macOS and Windows, the binaries are also code-signed for platform verification. In enterprise environments, you may want to verify signatures before installation.

Authentication: OAuth, API Keys, and Cloud Providers

Claude Code requires authentication to access Claude models. There are several authentication methods, each suited to different use cases.

OAuth (Claude.ai Subscription)

If you have a Claude Pro, Max, Team, or Enterprise subscription, you can authenticate using OAuth. This is the simplest method for individual users.

Run `claude` in your terminal. If you are not logged in, Claude Code will prompt you to sign in and open a browser window. Complete the authentication flow in the browser, and Claude Code will store your credentials locally.

This method ties usage to your subscription plan. Pro users get limited Claude Code usage included with their subscription. Max, Team, and Enterprise users get more generous limits. Usage beyond your plan limits may incur additional charges.

API Key

If you have an Anthropic Console account and an API key, you can use that to authenticate. This is useful for organizations that manage API usage centrally or for integrating Claude Code into automated pipelines.

Set the `ANTHROPIC_API_KEY` environment variable:

```
1  export ANTHROPIC_API_KEY=sk-ant-api03-xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

When you run `claude`, it will detect the API key and prompt you to approve using it for Claude Code. This creates a dedicated “Claude Code” workspace associated with your API key.

In production or automated environments, use your system’s secret management tool to inject the API key rather than hardcoding it.

Cloud Provider Integration

Claude Code can route API calls through cloud providers that host Claude models. This is relevant for organizations that use managed ML services.

- **Amazon Bedrock:** If your organization uses Bedrock, Claude Code can authenticate and route requests through it. This allows you to leverage existing Bedrock IAM policies and billing.
- **Google Vertex AI:** Similar integration for Google Cloud customers.
- **Microsoft Foundry:** Integration for Azure/Microsoft ecosystem customers.

These integrations require specific configuration and may have feature differences compared to direct Anthropic API access. For example, some Claude Code features like Cloud sessions may not be available through cloud provider integrations.

Keyless Authentication

As of version 2.1.243, the `/login` command in Claude Code offers a keyless authentication path using your Anthropic Console account directly. This is recommended for organizations that prefer not to manage API keys for Claude Code. It creates short-lived, scoped credentials instead.

Verifying Installation and Diagnostics

After installation and authentication, run through a quick verification to ensure everything is working correctly.

Basic Functionality Test

Launch Claude Code and run a simple command:

```
1  claude -p "What is 2 plus 2?"
```

This runs a non-interactive query and should return 4. If this works, the basic pipeline is functional.

Interactive Session Test

Start an interactive session:

```
1  cd /path/to/a/project
2  claude
```

Try a simple task like asking Claude Code to describe the project structure. This tests file access and interactive mode.

Diagnostics Command

Run the doctor command for a comprehensive check:

```
1  claude doctor
```

This checks:

- Binary version and integrity
- Authentication status
- Configuration files and their validity
- Available tools and permissions
- Known issues with your setup

Review the output carefully. If any checks fail, follow the suggested remediation steps.

Checking Active Configuration

Inside a Claude Code session, run `/status` to see your active configuration:

```
1  /status
```

This shows:

- Current model and effort level

- Loaded settings files and their precedence
- Active permissions rules
- Connected MCP servers
- Loaded project instructions

Use this regularly to understand what Claude Code is actually using at runtime.

Environment Considerations: Containers, WSL, Remote Systems

Claude Code is most commonly used on a developer's local machine. However, there are scenarios where you may want to run it in containers, WSL, or remote environments.

Development Containers

Running Claude Code in a development container (such as those managed by Codespaces, Gitpod, or Docker) is a valid approach. Benefits include:

- Consistent environments across team members
- Isolation of tools and dependencies
- Easy reproduction of development setups

Considerations:

- Ensure the container has the necessary prerequisites (bash, curl, etc.)
- Authentication must work in the container environment. OAuth browser flows may not work in headless containers.
- File permissions: Claude Code runs as the container user, so ensure it has appropriate access.

Example Dockerfile for a Claude Code-capable dev container:

```
1 FROM node:20-slim
2
3 RUN apt-get update && apt-get install -y \
4     bash \
5     curl \
6     git \
7     ripgrep \
8     && rm -rf /var/lib/apt/lists/*
9
10 RUN curl -fsSL https://claude.ai/install.sh | bash
11
12 ENV PATH="/root/.local/bin:${PATH}"
13
14 WORKDIR /workspace
```

WSL (Windows Subsystem for Linux)

Claude Code supports WSL 2. Installation and usage are the same as on native Linux. The native installer handles WSL automatically when invoked from a WSL terminal.

Considerations:

- File access: Claude Code can access files within the WSL filesystem efficiently. Accessing Windows-hosted files through `/mnt/c/` etc. works but may be slower.
- Git: Install Git for Windows or the WSL-native Git package for full Git functionality.
- Bash: WSL includes bash by default, so the Bash tool works without additional configuration.

Remote Systems and SSH

Running Claude Code over SSH is possible and sometimes useful. For example, when developing on a remote server or shared development environment.

Considerations:

- Terminal capabilities: Use a terminal emulator that supports the full TUI (terminal user interface) features Claude Code uses.
- Authentication: OAuth browser authentication does not work over SSH unless you have X11 forwarding or can access the browser on your local machine. API key authentication is more reliable for remote setups.
- Network: Ensure the remote system can reach Anthropic API endpoints.

Cloud Sessions and Cowork

Anthropic offers Cloud sessions, where Claude Code runs in an isolated VM managed by Anthropic rather than on your machine. This provides:

- Filesystem isolation: Claude Code cannot access your local filesystem beyond what is explicitly mounted.
- Network isolation: Outbound network access is controlled.
- Audit logging: All actions are logged and reviewable.

Cloud sessions require a claude.ai account and may not be available through all authentication methods (e.g., Bedrock, Vertex). They are primarily designed for enterprise environments where security and auditability are priorities.

Cloud sessions and Cowork do not read local device policies. They rely entirely on server-managed settings. This is intentional for security consistency.

Choosing an Environment

For most individual developers, a local installation on their workstation is the simplest and most flexible option. For teams, consider:

- Standardizing on a specific installation method
- Documenting authentication procedures
- Providing managed settings via a central policy for security consistency
- Using development containers or Cloud sessions for high-security environments

The environment you choose affects security, reproducibility, and team consistency. Make it deliberately, not by default.

Chapter 3: CLI Usage and Core Commands

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

Interactive Sessions and Terminal Integration

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

Non-Interactive Mode: One-Shot Execution and Scripting

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

Essential Flags: Model, Effort, Permission Mode, and Bare Mode

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

System Prompts and Customization

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

Continuation, History, and Session Management

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

Chapter 4: Configuration and Settings

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

Settings Files and Precedence

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

Permissions: Allow, Ask, Deny Rules

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

Environment Variables and Process Configuration

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

Model and Effort Configuration

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

Debugging Configuration and Troubleshooting

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

Chapter 5: CLAUDE.md and Project Instructions

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

CLAUDE.md Fundamentals and Scope

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

Writing Effective Project Instructions

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

Modular Rules and the .claude/rules Directory

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

Auto Memory and Persistent Context

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

Instructions Hierarchy and Conflict Resolution

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

Chapter 6: Tools and Capabilities

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

The Tool Architecture and Permission Model

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

Filesystem Tools: Read, Write, Edit, Glob, Grep

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

Shell Tools: Bash, PowerShell, and Command Execution

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

Web Tools: Search and Fetch

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

Agent Tools and Specialized Capabilities

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

Chapter 7: Agents and Subagents

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

Agent Spawning and the Task Tool

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

Subagent Configuration and Isolation

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

Parallel and Sequential Workflows

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

Dynamic Workflows at Scale

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

When Multi-Agent Overhead Is Worth It

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

Chapter 8: MCP and External Integrations

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

Understanding Model Context Protocol

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

Adding and Managing MCP Servers

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

Common Integrations: GitHub, Databases, APIs

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

Security and Credential Management for MCP

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

Build Your Own MCP Server

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

Chapter 9: Skills, Plugins, and Custom Commands

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

Skills: Structure and Invocation

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

Custom Slash Commands

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

Plugins and the Marketplace Ecosystem

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

Skill Evaluation and Testing

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

Chapter 10: Context Engineering

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

Context Windows and Token Economics

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

Information Prioritization Strategies

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

Repository Discovery and Codebase Indexing

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

Persistent vs. Transient Context

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

Stale Context and Its Dangers

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

Chapter 11: Prompt Design and Task Specification

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

Writing Specifications Claude Code Can Execute

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

Task Decomposition and Batching

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

Constraints and Guardrails

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

Verification Criteria and Acceptance Tests

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

Avoiding Ambiguity and Scope Creep

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

Chapter 12: Implementation Workflows

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

Feature Implementation from Requirements

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

Iterative Development and Feedback Loops

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

Refactoring with Confidence

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

Debugging and Incident Investigation

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

Handling Partial or Incorrect Implementations

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

Chapter 13: Testing and Quality Assurance

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

Test-Driven Development with Claude Code

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

Generating Tests: Unit, Integration, and E2E

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

Test Quality and Coverage

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

Static Analysis and Linting

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

Performance and Security Testing

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

Chapter 14: Git Workflows and Code Review

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

Branch Strategies and Claude Code

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

Commits and Conventional Messages

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

Pull Request Workflows

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

AI-Assisted Code Review

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

Maintaining Review Quality

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

Chapter 15: CI/CD and Automation

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

Claude Code GitHub Actions Setup

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

Automated PR Review and Response

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

CI Integration Patterns

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

Automation Safety and Rate Limits

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

Self-Hosted Runners and On-Premise Options

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

Chapter 16: Security and Governance

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

Permission Model and Least Privilege

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

Sandboxing and Execution Isolation

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

Secrets, Credentials, and API Keys

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

Prompt Injection and Adversarial Attacks

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

Supply Chain and Dependency Security

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

Data Privacy and Compliance

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

Security Summary Checklist

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

Chapter 17: Enterprise and Team Adoption

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

Managed Settings and Policy Enforcement

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

Team Conventions and Shared Instructions

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

Onboarding and Training

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

Collaboration Patterns

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

Measuring Adoption and Effectiveness

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

Chapter 18: Advanced Engineering Patterns

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

Working in Monorepos

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

Polyglot and Mixed-Technology Codebases

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

Legacy Code Modernization

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

Large-Scale Architectural Refactoring

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

Observability and Production Systems

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

Chapter 19: Case Studies and Workflows

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

Building a REST API Service from Scratch

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

Adding Authentication to an Existing Application

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

Migrating a Legacy Module to Modern Architecture

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

Investigating and Fixing a Production Incident

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

Chapter 20: Trade-offs, Limitations, and the Path Forward

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

Where Claude Code Excels

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

Where It Is Unreliable

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

Tasks That Demand Human Judgment

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

Cost, Complexity, and Long-Term Maintenance

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

The Future of Agentic Development

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

Honest Assessment

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

Conclusion: Engineering with Intelligence

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

References

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