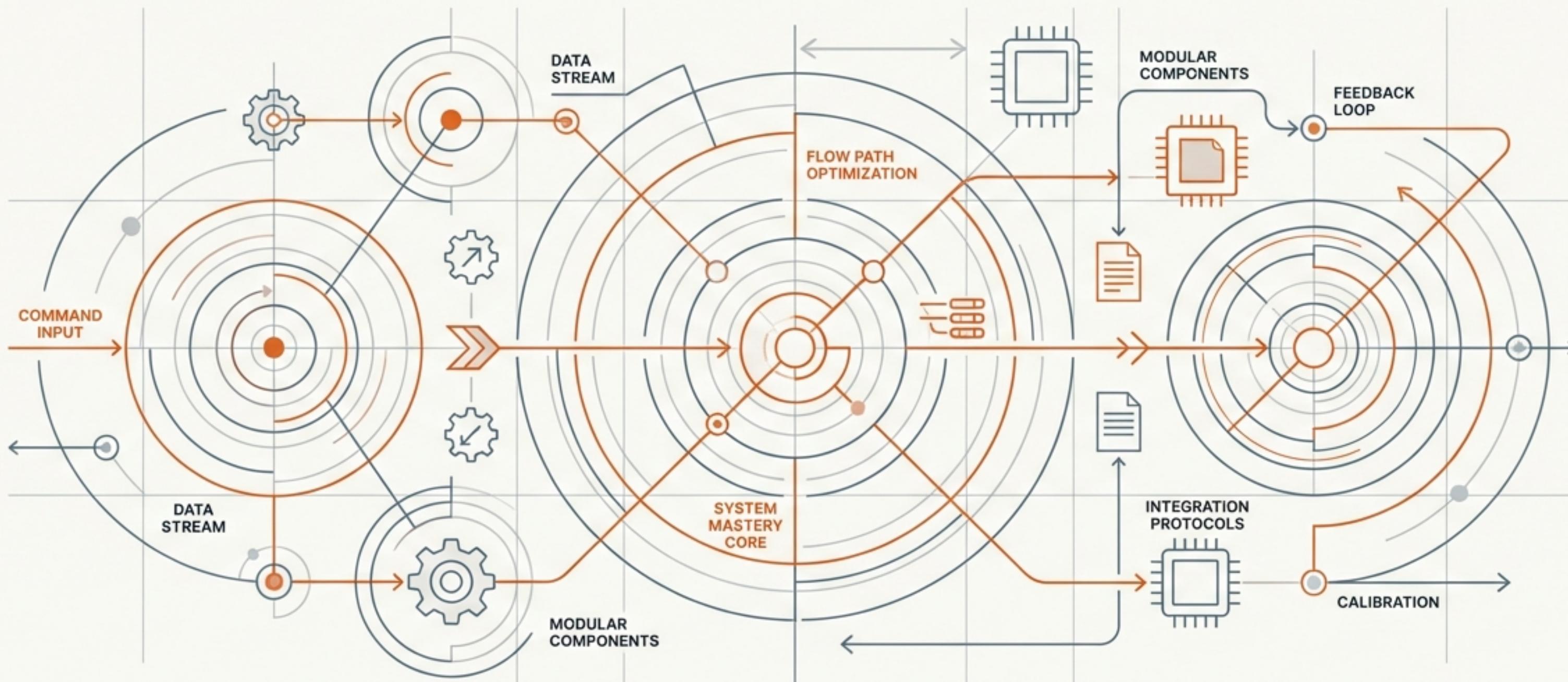


Master Your Copilot: A Developer's Guide to Claude Code

From your first command to full automation. This is your path to mastery.



The Core Interaction Loop: You Are Always in Control

4. You Decide

Review the diff, then Accept (y), Reject (n), or Edit (e).

3. Claude Proposes

Presents a diff of the proposed changes.

1. You Prompt

Describe the task, bug, or feature.

2. Claude Analyzes

Searches the codebase, identifies issues, and formulates a plan.



Your Day 1 Dashboard: Mission-Critical Commands

Essential Commands

Execute operations with precision.

	Command	Action
	<code>/help</code>	Contextual help
	<code>/clear</code>	Reset conversation
	<code>/compact</code>	Free up context
	<code>/status</code>	View token usage & session state
	<code>/exit</code>	Quit (or `Ctrl+D`)

Essential Shortcuts

Navigate and control rapidly.

	Shortcut	Action
	<code>Shift+Tab</code>	Cycle permission modes
	<code>Esc x 2</code>	Rewind (undo)
	<code>Ctrl+C</code>	Interrupt operation
	<code>Tab</code>	Autocomplete
	<code>@file.ts</code>	Reference a specific file

Setting the Rules of Engagement: Permission Modes

Default Mode (Safest)

- **Editing:** Asks Permission 
- **Execution:** Asks Permission

Best for

Learning the tool and for any unfamiliar tasks.

Auto-accept Mode (Fastest)

- **Editing:** Automatic
- **Execution:** Asks Permission



Use only for well-defined, reversible operations.

Plan Mode (Strategic)

- ✗ **Editing:** No
- ✗ **Execution:** No

Best for

Understanding unfamiliar code, exploring architecture, and safe investigation before making changes.

Press **Shift+Tab** to cycle between modes.

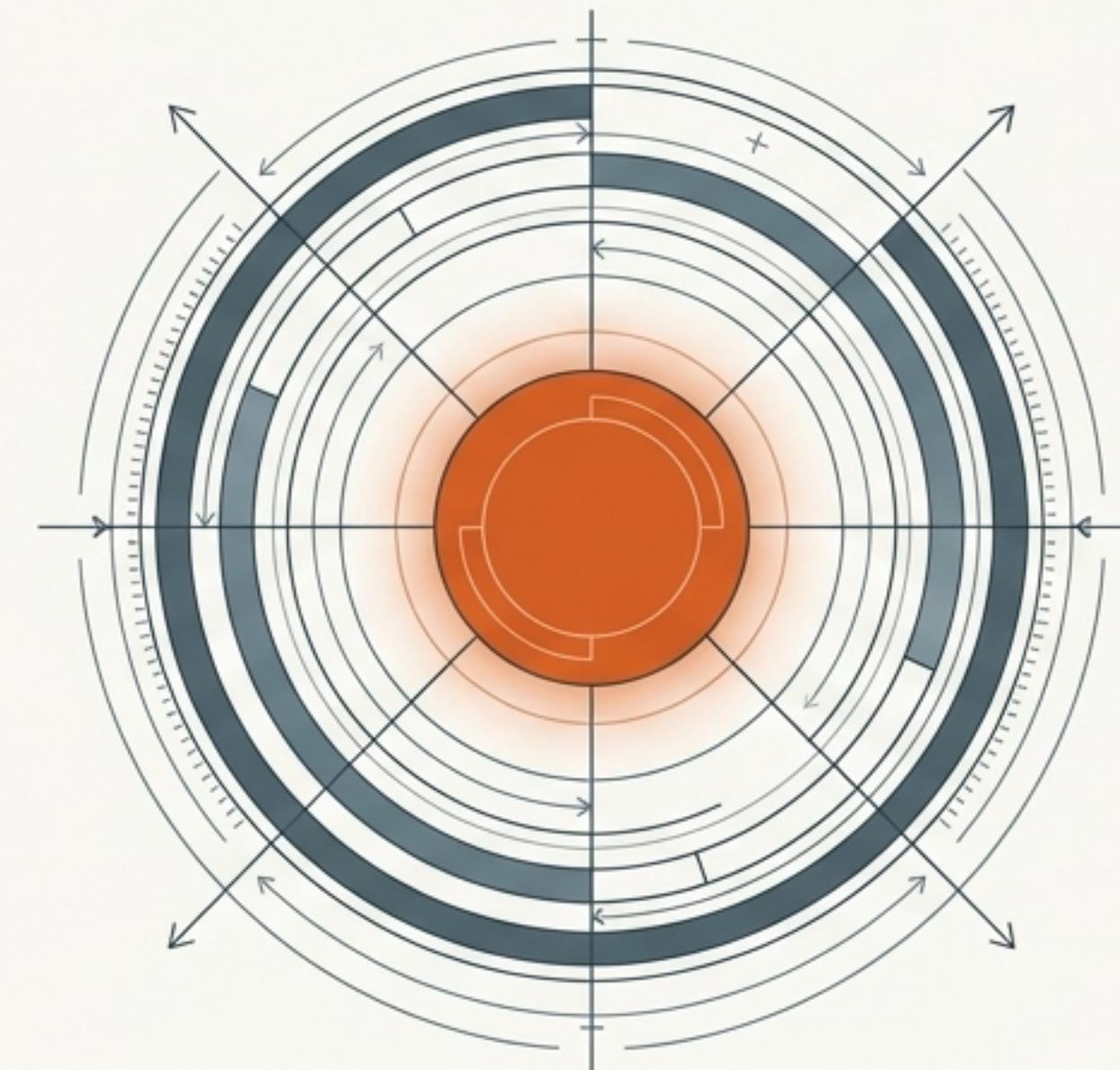
First Flight Complete: Your Day 1 Proficiency Checklist

- Launch Claude Code in your project.
- Describe a task and get a proposed change.
- Critically review the diff** before accepting or rejecting.
- Run a shell command directly using `!command`.
- Check your session state with `/status`.
- Start a new task cleanly with `/clear`.

Once you've mastered these, you're ready to understand the engine that powers it all.

The One Concept You Must Master: Context

Context is Claude's **working memory**. It's a finite resource of **200,000 tokens**. Managing it effectively is the key to unlocking the tool's full potential.



What is in Context?

- ⊕ Your conversation history
- ⊕ Files Claude has read
- ⊕ Outputs from commands and tools

Monitoring Your Context Fuel Gauge



Depletion Symptoms & Recovery

- ↗ Symptom: Short or incomplete responses.
→ Action: `/compact`
- ⚠ Symptom: Forgetting previous instructions.
→ Action: `/clear`
- ✂ Symptom: Errors on code already discussed.
→ Action: `New session needed.`

Pro-Level Context Hygiene

Avoid Context Poisoning

⌚ **Definition:** When information from one task contaminates another.

- **Common Patterns:** Style Bleeding (CSS from one component leaks into another), Instruction Contamination (old rules are misapplied to new tasks).

Solution: Use `/clear` between distinct, complex tasks.

The Sanity Check Technique

⌚ **Purpose:** A quick way to verify that your `CLAUDE.md` instructions have been loaded correctly.

- **How-to:**

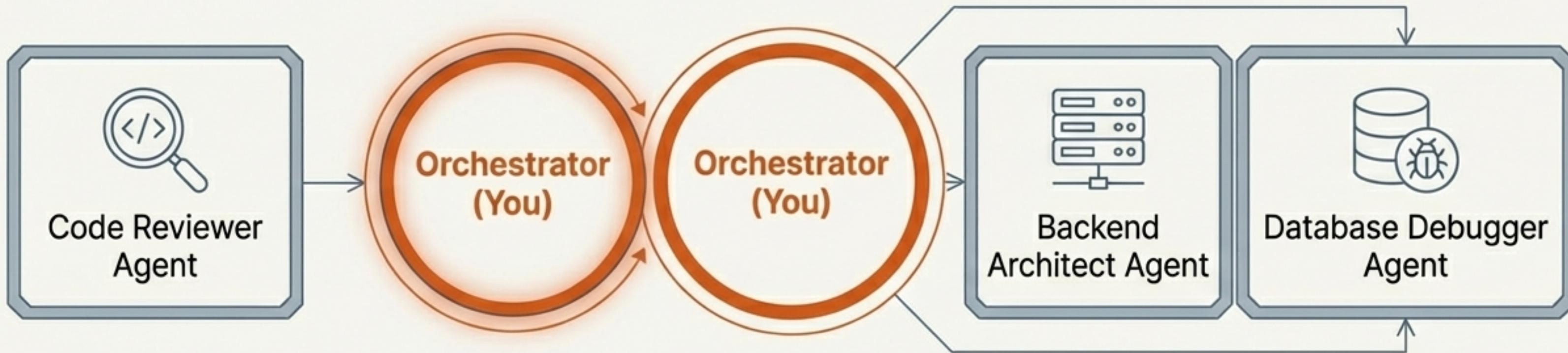
- 1 Add a simple checkpoint to your `CLAUDE.md`: `<!-- CHECKPOINT: My name is [Your Name]. -->`
- 2 Ask Claude at the start of a session: "*What is my name?*"

Troubleshooting Table

Failure Symptom	Probable Cause	Solution
Doesn't know your name	<code>CLAUDE.md</code> not loaded	Check file location
Partial knowledge	Context exhausted	<code>'/clear'</code> and retry

Unleashing Full Potential: Extending Your Cockpit with Agents

Agents are specialized AI personas you create to delegate specific tasks. Instead of one generalist, you command a team of experts.



Without Agents	With Agents
One Claude doing everything	Specialized experts for each domain
Context gets cluttered	Each agent has focused context
Generic responses	Domain-specific, expert responses

Building Your Toolkit: Agents, Skills, and Commands

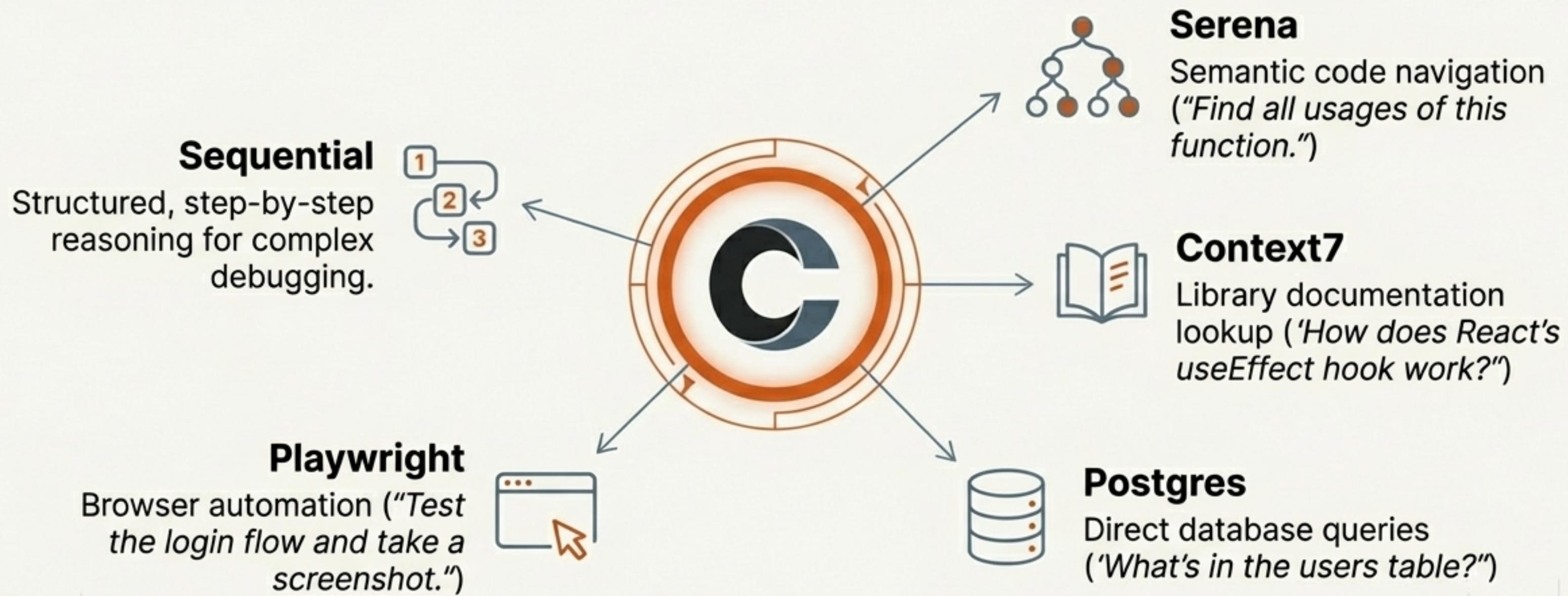
Concept	Purpose	Invocation
Agent	A specialized role to delegate tasks to	Activated automatically by your prompt
Skill	A reusable knowledge module for agents to inherit	Inherited by an agent via its config
Command	A scripted process workflow for repetitive tasks	Manually run via a /slash command

```
.claude/agents/reviewer.md
```

```
name: code-reviewer
description: Proactively reviews code for style, errors, and best practices.
tools: [mcp_serena_find_symbol]
skills: [security-guardian-skill]
```

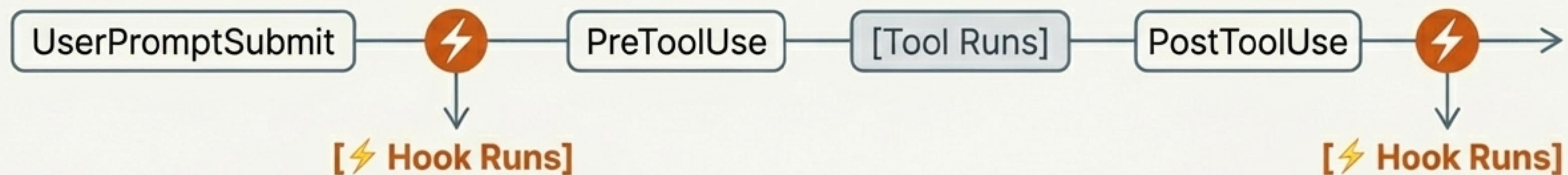
Connecting to External Systems: MCP Servers

MCP (Model Context Protocol) connects Claude to external tools, giving it capabilities far beyond simple file I/O.



True Automation: Triggering Actions with Hooks

Hooks are scripts that run automatically when specific events occur, letting you build guardrails and automate your workflow.



Practical Example: The Linting Gate

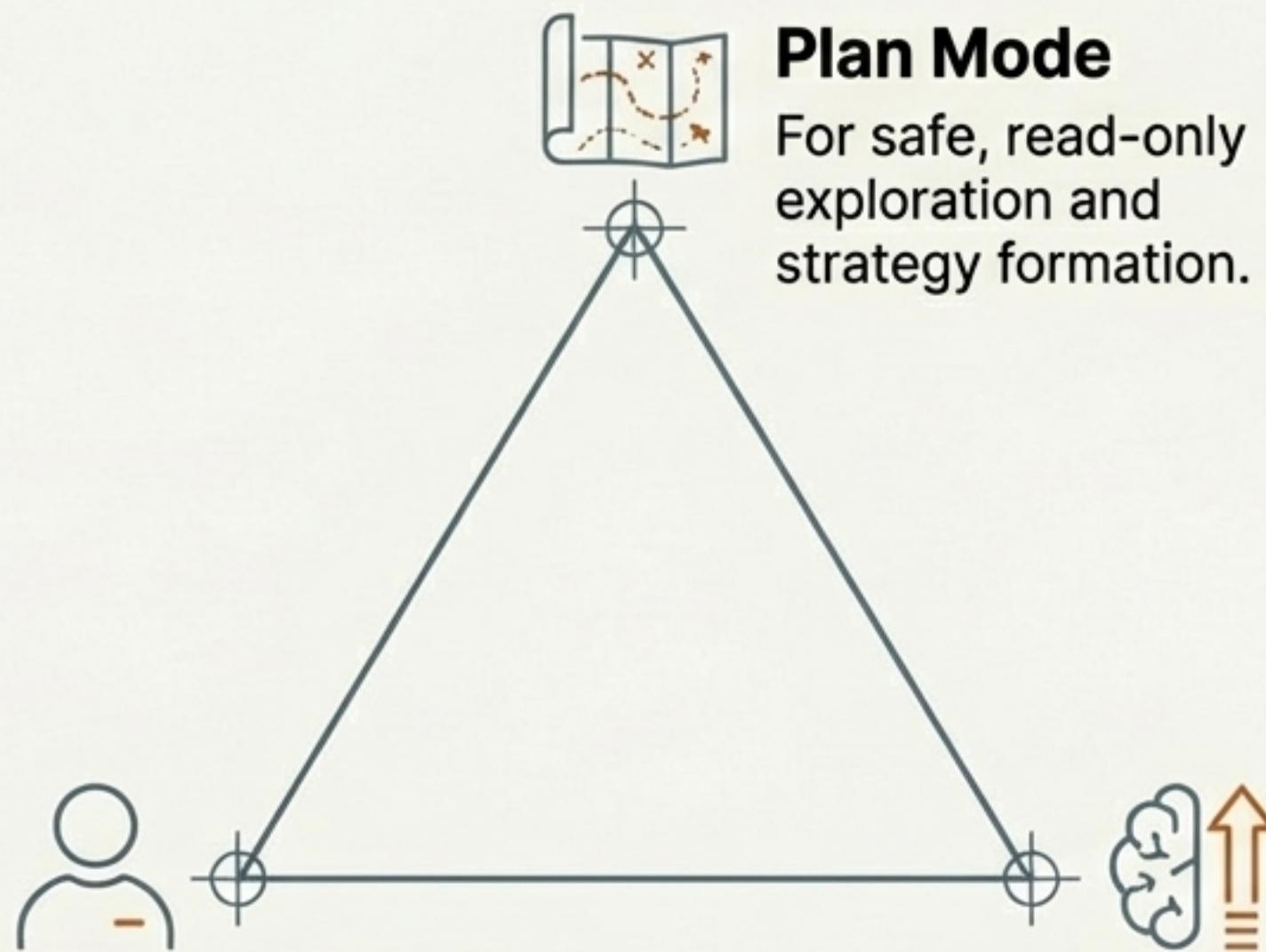
Goal: Prevent Claude from committing code that fails the linter.

Event: PreToolUse

Matcher: Bash(git commit *)

Action: A script (.claude/hooks/lint-gate.sh) runs pnpm lint. If it fails, the hook exits with code 2, blocking the commit and informing the user.

The Power User Pattern: The Trinity



Specialized Agent

To bring domain-specific expertise (e.g., Backend Architect).

Ultrathink

To allocate a massive token budget for deep, multi-file reasoning.

Ultrathink Levels

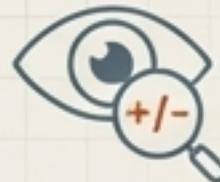
Flag	Thinking Depth	Token Usage	Best For
--think	Standard	~4K	Multi-component analysis
--think-hard	Deep	~10K	Architectural decisions
--ultrathink	Maximum	~32K	Critical redesign, legacy modernization

Use the Trinity for high-stakes tasks: complex debugging, architectural planning, and modernizing legacy systems.

Your Path to Mastery: The Claude Code Maturity Model

Level	Name	Key Characteristics	Typical Timeline
5	Expert	Integrates into CI/CD, builds custom tooling for the team.	Month 2+
4	Advanced	Masters MCP servers, orchestrates multi-agent workflows.	Month 1-2
3	Proficient	Creates custom Agents, uses Plan Mode strategically, basic Hooks.	Week 2-4
2	Competent	Reviews all diffs, uses `/compact`, has a project `CLAUDE.md`.	Week 1-2
1	Beginner	Uses basic commands, minimal config.	Day 1-7

The 6 Golden Rules of the Expert Pilot



- 1. Always review diffs** before accepting. You are the final authority.
- 2. Use /compact before context gets critical (>75%).** Maintain situational awareness.
- 3. Be specific** in requests. Provide WHAT, WHERE, and HOW. Clear communication is key.
- 4. Use Plan Mode first** for complex or risky tasks. Reconnaissance before action.
- 5. Create a `CLAUDE.md`** for every project. Define the mission parameters upfront.
- 6. Commit frequently** after each small, completed task. Create save points.

