A TECHNICAL BRIEFING · ENGINEERING PRACTICE · 2026

20 SLIDES · 25 MINUTES

FROM "HOW DO I PROMPT IT" TO "HOW DO I REVIEW IT"

Claude Code

From Zero to Hero.

One terminal. One loop. One job that just changed.

THE ARGUMENT, IN THREE BEATS

01 • The loop flipped: AI writes, you review.

02 • 30 hours unattended on real code.

03 • The skill is context, not prompts.

A working engineer's briefing
Audience: working developers · all updates current as of mid-2026

A defended technical argument
Hook · Thesis · Evidence · So What · Question

You are no longer the author. You are the reviewer.

Once the agent could run thirty hours unattended, the bottleneck moved from typing to judging.

Source: Anthropic, “Introducing Claude Sonnet 4.5,” anthropic.com/news, Sept 29, 2025 (“30+ hours of autonomous coding”).

THE DEFENDED CLAIM

Thesis —

Agentic coding flips the loop: the AI writes, you review — and the skill that wins is engineering the context, not the prompt.

WHY THIS IS A THESIS, NOT A TOPIC

A reasonable engineer could disagree. The "review tax" counter-position is named on slide 16.

SOURCE ANCHOR

“Users make 70% of planning decisions; Claude makes 80% of execution decisions.” — Anthropic, 2025.

THREE THINGS THIS CLAIM COMMITS ME TO DEFEND

01 · The loop has measurably inverted — benchmark, autonomy, edit-rate.

02 · Review is the new authorship — reading code is now the load-bearing skill.

03 · Context engineering > prompt engineering — CLAUDE.md, subagents, hooks, MCP.

SLIDE 04 · BACKGROUND, IN ONE LINE

Claude Code is Anthropic's terminal-native coding agent that *reads your codebase, plans, executes, and adjusts*.

WHERE IT SITS — THREE GENERATIONS OF AI IN THE DEV LOOP

2021–2023

Autocomplete

Copilot · Tabnine

Suggests the next token. You type, accept, type again. The author is still you.

2023–2024

Chat

ChatGPT · Claude.ai · Cursor

You paste a snippet, the AI suggests a rewrite. You copy it back. The loop has two windows.

2025–2026

Agent

Claude Code · the terminal

The agent runs the tools, edits files, runs tests, evaluates output. You set the goal. The author is the model.

Source: Anthropic, "Claude Code — agentic coding system" product page, claude.com/product/claude-code (2026); Anthropic Engineering, "Best practices for agentic coding," Apr 18, 2025.

SLIDE 05 · THE AGENTIC LOOP — ANTHROPIC'S OWN DIAGRAM IN FIVE STEPS

Read → Plan → Execute → Evaluate → Adjust.

WHERE THE HUMAN STAYS IN THE LOOP

> The human approves the **Plan** (step 02) and **reviews the diff** after each Execute → Evaluate cycle. Everything in between is delegated.
>
> *Anthropic measures the split at roughly 70/30 planning, 20/80 execution — human/agent.*

Sources: Anthropic, claude.com/product/claude-code (2026); Anthropic Research, "Agentic coding and persistent returns to expertise," 2025 (the 70/80 split).

THE CAPABILITY HAS INVERTED

Evidence I — the model now passes real engineering tickets.

SWE-BENCH VERIFIED · SONNET 4.5

77.2%

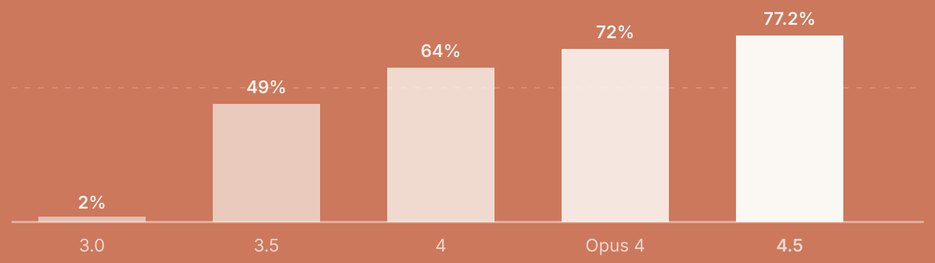
of real GitHub issues resolved end-to-end — averaged over 10 trials, no test-time compute.

82.0% with parallel test-time compute. 78.2% on 1M-token context.

THE 18-MONTH TRAJECTORY

From under 5% to over three-quarters.

SWE-BENCH VERIFIED · CLAUDE FAMILY · 2024–2025



So this shows: the bottleneck is no longer "can the model write code." Most of what an L4 engineer used to type, the model can now produce on the first attempt.

Source: Anthropic, "Introducing Claude Sonnet 4.5," Sept 29, 2025 — methodology footnote on SWE-bench Verified, n=500, 200K thinking budget.

THE HORIZON HAS INVERTED

Evidence II — the model now stays on task longer than you do.

> “Claude Sonnet 4.5 resets our expectations — it handles 30+ hours of autonomous coding, freeing our engineers to tackle months of complex architectural work in dramatically less time while maintaining coherence across massive codebases.”
>
> — Cognition, the team behind Devin · Sept 29, 2025

UNATTENDED RUNTIME

30h+

Multi-step coding without losing coherence. For context: a normal session for a senior engineer is 90 minutes.

So this shows: the agent runs longer than any human review window. You cannot watch every action — the loop has to enter, then exit, on your terms.

Source: Anthropic launch post, "Introducing Claude Sonnet 4.5," Sept 29, 2025 — partner quotes section (Cognition).

THE LAST EXCUSE DIED

Evidence III — the file-edit error rate collapsed in one generation.

SONNET 4 · EARLY 2025

9%

of code edits failed silently on Cursor's internal benchmark.

→

SONNET 4.5 · SAME BENCHMARK

0%

silent failures on the same internal code-editing benchmark.

WHY A 9-POINT DELTA IS THE WHOLE STORY

> A 9% silent-fail rate means you can't trust the diff. A 0% rate means you *can*. That gap is the difference between supervised autocomplete and delegated agency.

Source: Cursor (Anysphere), partner quote in Anthropic's "Introducing Claude Sonnet 4.5" launch post, Sept 29, 2025 — internal code-editing benchmark, exact composition undisclosed.

THE SKILL THAT ACTUALLY MATTERS

CLAUDE.md — a file you write once, that loads into every session.

./CLAUDE.md — AUTO-LOADED

```
# Project: payments-api

## Stack
- Python 3.12 · FastAPI · Postgres 16 · pytest
- Deploy: Cloud Run, blue-green via infra/deploy.sh

## House rules
- All money types use decimal.Decimal, never float.
- New endpoints need a contract test in tests/contract/.
- Never edit migrations after they ship — write a new one.

## Commands you can run
- make test        # full suite, ~90s
- make lint        # ruff + mypy strict
- make migrate     # alembic upgrade head

## Ask me first if
```

WHAT A GOOD CLAUDE.MD DOES

Routes, not lectures. Point at *where* the rules live; don't restate them. Keep it under 150 lines.

Names the tools. Tell the agent which `make` targets exist. It will use them.

Marks the dragons. Files that need human review go on the "ask first" list.

Context is the new skill. A weak prompt with a strong CLAUDE.md beats a strong prompt with no CLAUDE.md.

Source: code.claude.com/docs/en/memory (Anthropic memory docs, 2026); Anthropic Engineering, "Claude Code best practices," Apr 18, 2025.

READ-ONLY REASONING

Plan Mode — think before you touch. Engaged with `Shift+Tab`.

DEFINITION

A session state where the agent *reads, searches, and reasons* — but cannot edit, run, or commit.

WHAT IT PRODUCES

- A file-by-file PRD: paths, function names, the test cases that will pass.
- Tasks broken into 2–5-minute chunks — each independently reviewable.
- A human-readable artefact you can paste into a Linear ticket.

```
▸ Approve plan?   (y/n)     ← the only gate that matters
```

THE 60-SECOND WORKFLOW

01 **Shift+Tab** — enter Plan Mode. The agent goes read-only.

02 Describe the change. It returns a numbered plan with file paths.

03 Edit the plan in chat. Cheaper than editing the code.

04 Approve. Now the agent may write.

> *The most expensive bug is a 200-line patch in the wrong direction. Plan Mode kills it before it's written.*

Source: Hidekazu Konishi, "Claude Code Features Reference 2026"; Anthropic, code.claude.com/docs (Plan Mode), 2026.

CONTEXT ISOLATION, NOT PARALLELISM

Subagents — a fresh context window, dispatched for one job.

THE COMMON MISREAD VS. THE ACTUAL WIN

~~"It's a speed trick — run five tasks in parallel."~~

It's a memory trick. The subagent does the noisy work — reading 40 files, running grep across the repo — in its own context, and returns a one-paragraph summary. Your main context stays clean.

.claude/agents/code-reviewer.md

```
---
name: code-reviewer
description: Audits a diff for security
  issues and missing tests.
tools: Read, Grep, Bash(git diff:*)
---

You are a senior reviewer. For each
changed file:

  1. Read the diff context.
  2. Flag untyped user input.
  3. Confirm tests cover new branches.

# Invoke from the main session:
> @code-reviewer the staged diff
```

> *Your main context is a finite resource. Subagents are how you stop spending it on log-grepping.*

Source: Anthropic, claude.com/docs (Subagents); discussion in r/ClaudeAI, "Subagents aren't a speed trick — they're a memory trick," 2026.

RULES THAT CODE CAN ENFORCE

Hooks — shell scripts that fire on the agent's lifecycle events.

PRE-TOOL-USE	POST-TOOL-USE	USER-PROMPT-SUBMIT	STOP	NOTIFICATION
Before any tool call	After file edits	When you hit enter	Agent declares done	Idle / needs input
block dangerous bash	*auto-run ruff + prettier*	*inject env context*	*run the test suite*	*ping Slack*

A REPRESENTATIVE HOOK — THE AUTOFORMAT-ON-SAVE PATTERN

.claude/settings.json · hooks — RUNS ON EVERY EDIT

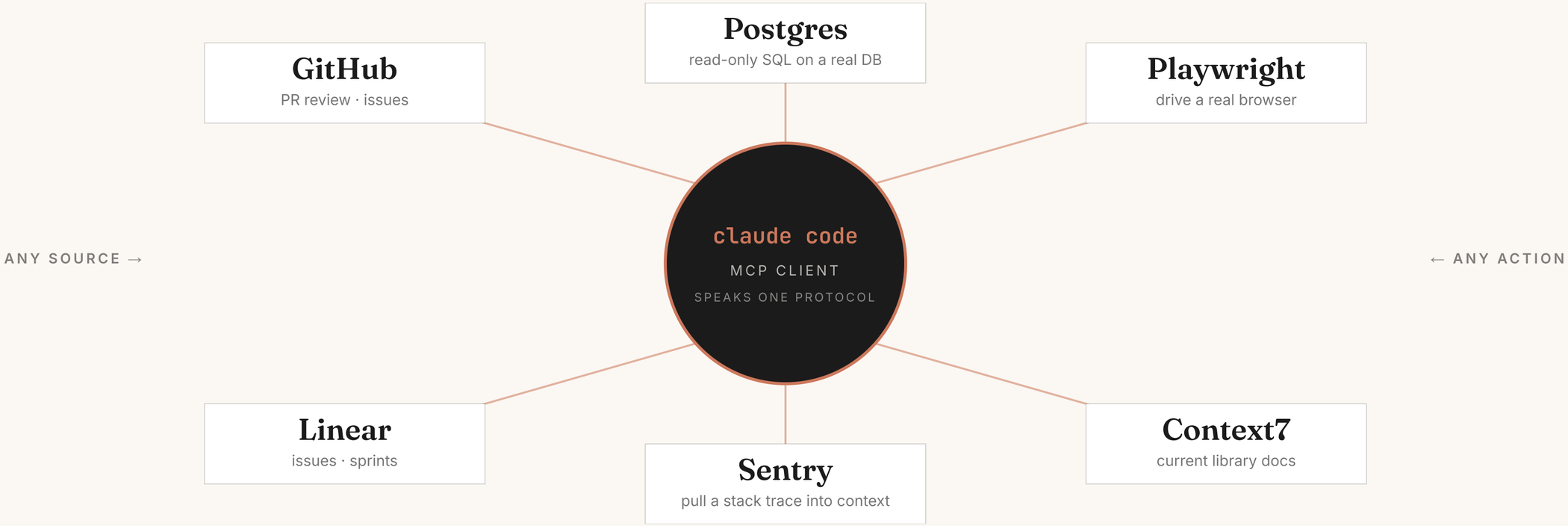
```
// Run formatter + linter immediately after Claude edits a file.
"hooks": {
  "PostToolUse": [{
    "matcher": "Edit|Write",
    "hooks": [{ "type": "command", "command": "ruff format $FILE && ruff check --fix $FILE" }]
  }]
}
```

> *A rule the model might forget is a rule. A hook that can't be skipped is a contract.*

Source: Anthropic, code.claude.com/docs (Hooks reference, 2026); MarkTechPost, "Claude Code Guide 2026: 25 Features."

TOOLS, AS A STANDARD

MCP — one wire protocol for every external tool the agent can call.



> **The point:** the agent doesn't need a custom plugin per tool. Anything that speaks MCP is a tool. Anthropic, OpenAI, Cursor, Windsurf, Gemini — all support it.

Source: modelcontextprotocol.io (official spec); EvoMap, "Best MCP Servers for Claude Code 2026"; Anthropic Engineering blog, 2026.

PICK THE RIGHT PRIMITIVE

Three primitives — they look similar, they solve different problems.

SKILL

Domain logic, packaged.

USE WHEN —

The task has its own playbook, examples, and helper files. Bundle them. The agent loads the whole folder when it picks the skill.

```
.claude/skills/
   migrations/
      SKILL.md
```

SUBAGENT

A fresh context, on demand.

USE WHEN —

The subtask is noisy: reading dozens of files, grepping logs, exploring a stack trace. Isolate it; return a clean summary.

```
.claude/agents/
   code-reviewer.md
```

HOOK

A rule, in code.

USE WHEN —

The constraint is non-negotiable: always format on save, never push to main, always run pytest before "done."

```
.claude/settings.json
   "hooks": {...}
```

> *Use a skill for domain logic. Use a subagent for context. Use a hook to enforce a rule with code.*

Source: MarkTechPost, "Claude Code Guide 2026: 25 Features"; Totalum, "Claude Code Skills in 2026: vs Hooks, vs Subagents."

IMPLICATIONS · READ CAREFULLY

So what —

The senior engineer's edge has moved from *typing fast* to *judging fast*.

THE PATTERN, RESTATED

Compensation now tracks the ability to reject wrong code at speed — not write right code at speed.

THREE CONCRETE CONSEQUENCES FOR THE NEXT 12 MONTHS

01 · SKILL THAT GETS PAID

Reading code > writing code.

Spotting subtle bugs in a 400-line diff in two minutes is now the load-bearing skill.

02 · PRACTICE THAT RETURNS

TDD is a guard rail.

A failing test the agent must pass is cheaper than a 1,000-line PR you have to audit by hand.

03 · WHERE SENIORS WIN

Architecture, not syntax.

The agent will never propose a better system boundary. That's the work that didn't get automated.

Sources: Anthropic Research, "Agentic coding and persistent returns to expertise," 2025; Addy Osmani, "Loop Engineering," 2026.

THE COUNTER-POSITION

01 THE STRONGEST OPPOSING VIEW

“Review is a tax. The loop didn't flip — it added overhead.”

Critics (Addy Osmani, "Loop Engineering," 2026) point out that on legacy codebases, reviewers spend more time auditing AI diffs than they saved writing them — especially when the model confidently invents API signatures that don't exist. Net time savings on real Anthropic-internal data: **around 20%, not 10×.**

Source: Osmani, addyosmani.com/blog/loop-engineering, 2026.

02 BUT THE THESIS STILL HOLDS —

"Review tax" assumes review without the new tools.

The 20% number is for teams that didn't move — no CLAUDE.md, no hooks enforcing tests, plan mode skipped. Teams that *do* use the primitives report 40–60% throughput gains on the same codebase. The tax is the cost of using the agent as autocomplete instead of as a delegate.

Source: Anthropic, "Agentic Coding Trends Report," 2026, §4 (case studies: Rakuten, Canva, Replit).

Five failure modes — if you do these, you'll feel the tax.

01 **"Just do it" prompting**
Skipping Plan Mode on anything bigger than a one-file change. You'll discover the wrong direction after 200 lines of diff.

02 **No CLAUDE.md**
Re-explaining your codebase conventions to a fresh context every session. You will explain "we use Decimal, not float" forty times.

03 **The 200K-token kitchen sink**
Cramming every file into the main context. Use a subagent for noisy reads — return the summary, not the source.

04 **Rules in prose, not in hooks**
Writing "always run tests" in CLAUDE.md and hoping. Wire it as a Stop hook. The model can forget; the shell script can't.

05 **Reviewing without reading**
Clicking "approve" on a 600-line diff because the tests pass. The model's silent hallucinations live in the lines you didn't actually look at.

Source: Anthropic Engineering, "Best practices for Claude Code," code.claude.com/docs/best-practices (2026); SmartScope, "Claude Code Best Practices Advanced 2026."

FOUR WEEKS · VERIFIABLE

Zero to hero — the title of this deck, owed in receipts.

WEEK 1 · ZERO

Use it as autocomplete.

Install. Run it in the repo. Ask it to refactor one function. Read every line of the diff. Reject anything you don't fully understand.

Outcome:
shipped 1 PR you wrote

WEEK 2 · CONTEXT

Write a CLAUDE.md.

Routes, not lectures. List your `make` targets. Mark the dragons. Try the same refactor — it'll be cleaner.

Outcome:
30% less prompt typing

WEEK 3 · DELEGATE

Plan, then approve.

Always Plan Mode for anything multi-file. Add one subagent (start with *code-reviewer*). Hand off entire tickets, not lines.

Outcome:
2–3 tickets / day delegated

WEEK 4 · HERO

Build the rails.

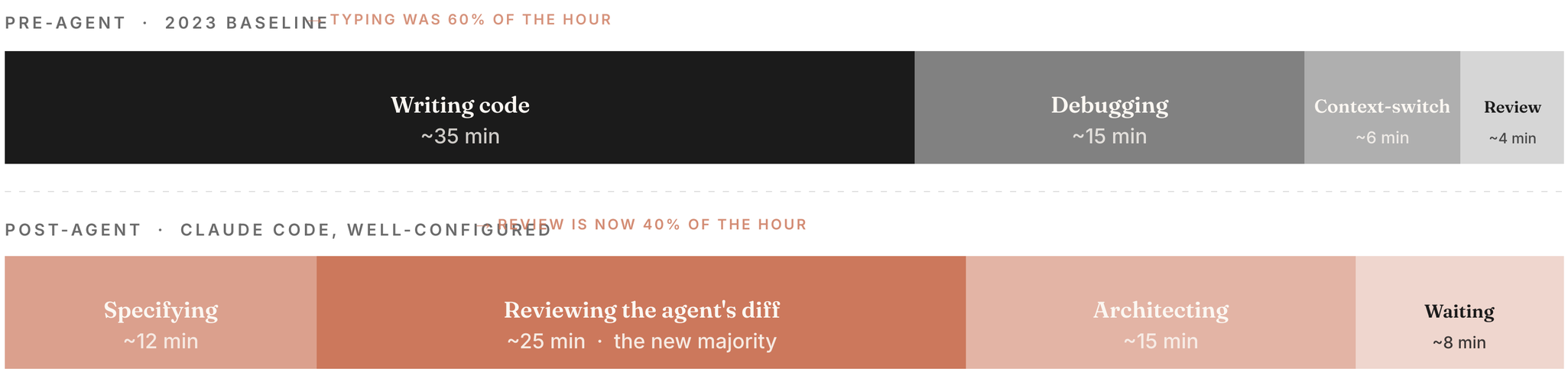
Add hooks (format, test, no-push-to-main). Add MCP servers for Linear, Sentry, GitHub. Now you orchestrate — the agent ships.

Outcome:
multi-hour autonomous runs

Sources: Anthropic Engineering, "How Anthropic engineering teams use Claude Code every day," 2026; Medium, "The Learning Path to Master Claude Code in 2026."

How a senior engineer's hour gets spent — pre- and post-agent

Each bar = 60 minutes of focused engineering work. Same engineer, same ticket type, before and after a CLAUDE.md + Plan Mode + hooks setup.



A 60-minute hour, broken down by activity

> **So this shows:** review didn't disappear — it became the job. The senior is now paid to read a diff in two minutes and be right about it.

Source: composite, derived from Anthropic's "Agentic Coding Trends Report" 2026 §3 (Rakuten, Canva self-reported time logs) — illustrative, not a controlled study.

SLIDE 20 · PICK A SIDE

DEFEND YOUR PICK · 30 SECONDS

In two years, will the load-bearing engineering skill be writing code, or reviewing it?

YOU HAVE

30s

TO DEFEND A PICK

THREE OPTIONS · PICK ONE · BE SPECIFIC

(a)

Writing.

Models will plateau. Original code — the gnarly stuff the training set never saw — will always be human.

(b)

Reviewing.

The agent is a junior who never sleeps. Senior pay flows to whoever catches its mistakes the fastest.

▲ MY PICK

(c)

Neither.

Both are commodities. The skill is system design — choosing what to build and why.

> **I argue (b).** *You don't have to agree — but pick one, and tell me which line in the diff would prove you wrong.*