

Context Engineering for LLM Coding



Hi, I'm Toby! You might know me from ...



2004



2014



FRONTASTIC.CLOUD
Frontastic closes Pre-Series-A over 1.8 million euros -
FRONTASTIC – agile Frontend as a Service

2019

- Web since 1996, PHP since 2000
- ~~Principal Engineer at commercetools Frontend~~
- Sabbatting, projects in the pipeline, talk to me :)

AI slop pull requests overwhelm open source projects.

The Reverse Centaur

- Fuzzy instructions → LLM pollutes its own context
- Little room left for actual code generation
- Low-quality, disconnected output

You end up reviewing AI code all day.

Being the Centaur



**Be deliberate about where human judgment ends
and machine execution begins.**

The Centaur Model

- **You own**: intent, scope, constraints, trade-offs
- **The LLM assists**: researching, structuring, planning
- **The LLM executes**: writing the actual code
- Context is **deliberately crafted**, not discovered on the fly

The Workflow

Four Phases, Four Artifacts

- `/create_ticket` → `ticket.md`
- `/research_codebase` → `research.md`
- `/create_plan` → `plan.md`
- `/implement_plan` → code + tests + commits

Each phase runs in a **fresh context** (`/clear` between phases)

Define & Research

/create_ticket

- Clarify what to achieve
- Define edge cases and scope
- LLM asks probing questions
- → Focused **ticket document**

/research_codebase

- Takes ticket as input
- Finds files, patterns, side-effects
- Spawns **parallel agents**
- → Detailed **research document**

Plan & Implement

/create_plan

- Reviews ticket + research
- **Human picks** solutions
- Step-by-step TODO list
- Includes test specifications

/implement_plan

- Reads all three Markdown files
- Executes plan step by step
- Writes code, runs tests
- Commits in logical phases

Supporting Infrastructure

Agents

- **codebase-analyzer** — understand mechanisms end-to-end
- **codebase-pattern-finder** — identify patterns for consistency
- **web-search-researcher** — fetch docs, best practices, known issues

Each runs in its **own isolated context**

Thoughts Directory

- Every command result stored as Markdown in `thoughts/`
- Organized: `tickets/`, `research/`, `plans/`, `reviews/`
- Per-user or `shared/` for team-wide knowledge
- **Grows into a searchable knowledge base**

CLAUDE.md & Hooks

- **CLAUDE.md** files document structure and rules throughout the repo
- Main file: project overview, component map, general rules
- Component-level files: specific conventions
- **Hooks**: quality gates, commit validation, notifications

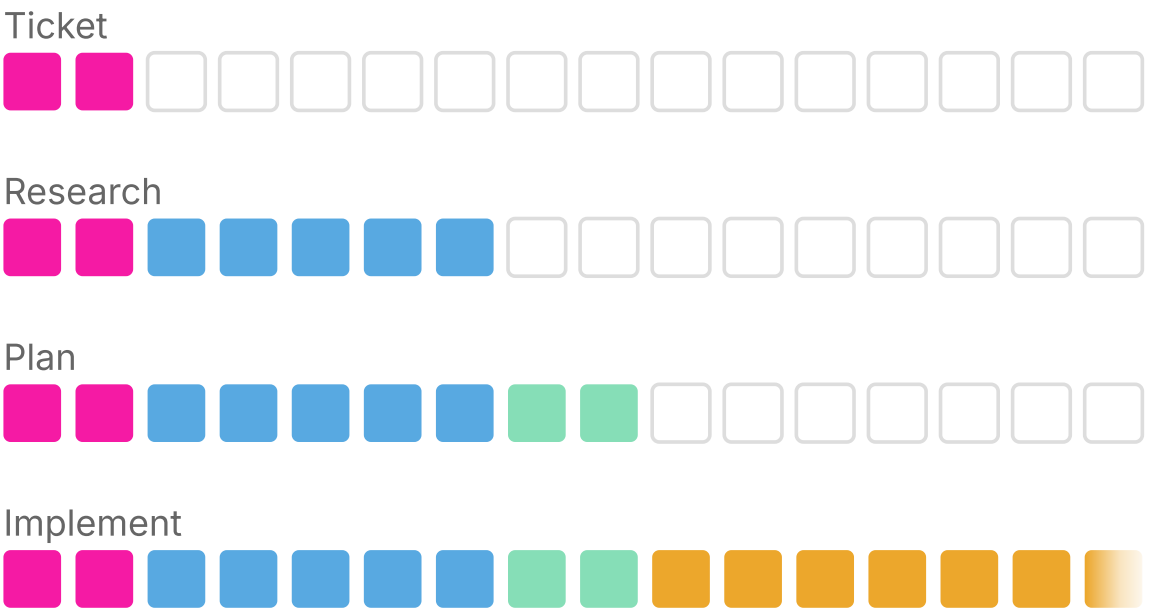
Supporting Commands

- **/discuss** — explore topics, structure thinking, produce discussion documents
- **/code_review** — validate against codebase patterns and constraints
- **/commit** — enforce explicit commit points between phases

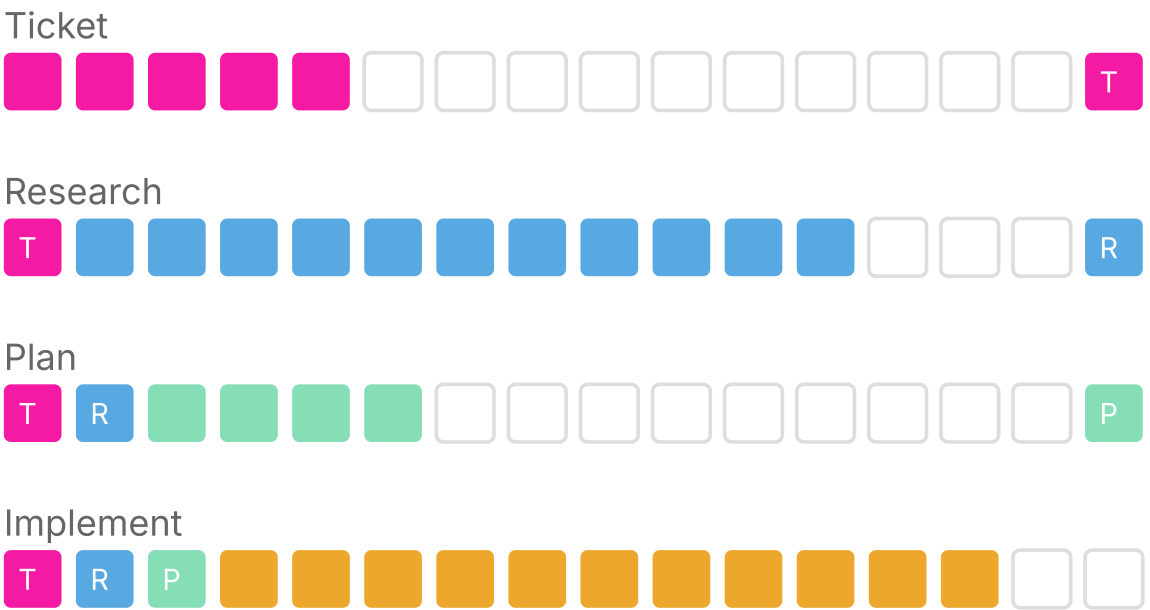
Why It Works

Context Window Usage

Single Session



Phased Context Engineering



Why It Works

- **Token efficiency** — small, focused context → more room for code
- **Human steering** — every architectural decision has an owner
- **Compounding knowledge** — thoughts directory grows smarter with every ticket

The LLM does not replace the thinking — it accelerates the translation of already-formed ideas into concrete code.

Links

- Blog post on context engineering:
schlitt.info/blog/0793_context_engineering_claude_code.html
- Context engineering template for Claude Code:
github.com/tobyS/claude-template

Questions?



tobias@schlitt.info
 Schlitt.info
[linkedin.com/in/tobiasschlitt](https://www.linkedin.com/in/tobiasschlitt)

