# Superpowers

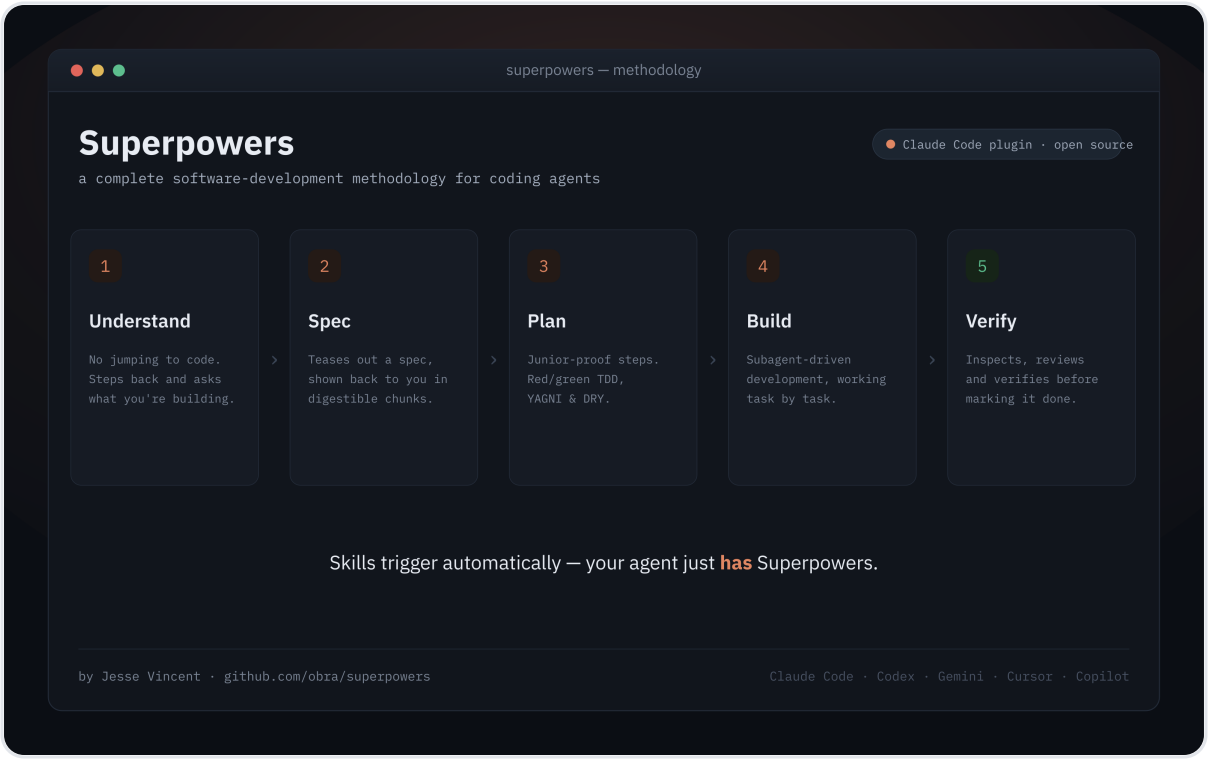
A complete software-development methodology for coding agents

Open-source plugin by Jesse Vincent · github.com/obra/superpowers

Superpowers isn't another autocomplete wrapper. It bolts a complete development methodology onto your coding agent as a library of composable skills that trigger automatically — so the agent specs and plans before it writes code, and you stay in control at every gate.

### The problem it solves

Developers rarely write a complete specification — a corner case gets missed, a question goes unasked, and it surfaces later in production. Superpowers closes that gap up front. It asks the relevant questions, exposes the scenarios you'd otherwise discover the hard way, and turns the answers into a strong plan — one you review and approve before a single line is executed.




The Superpowers workflow: understand → spec → plan → build → verify

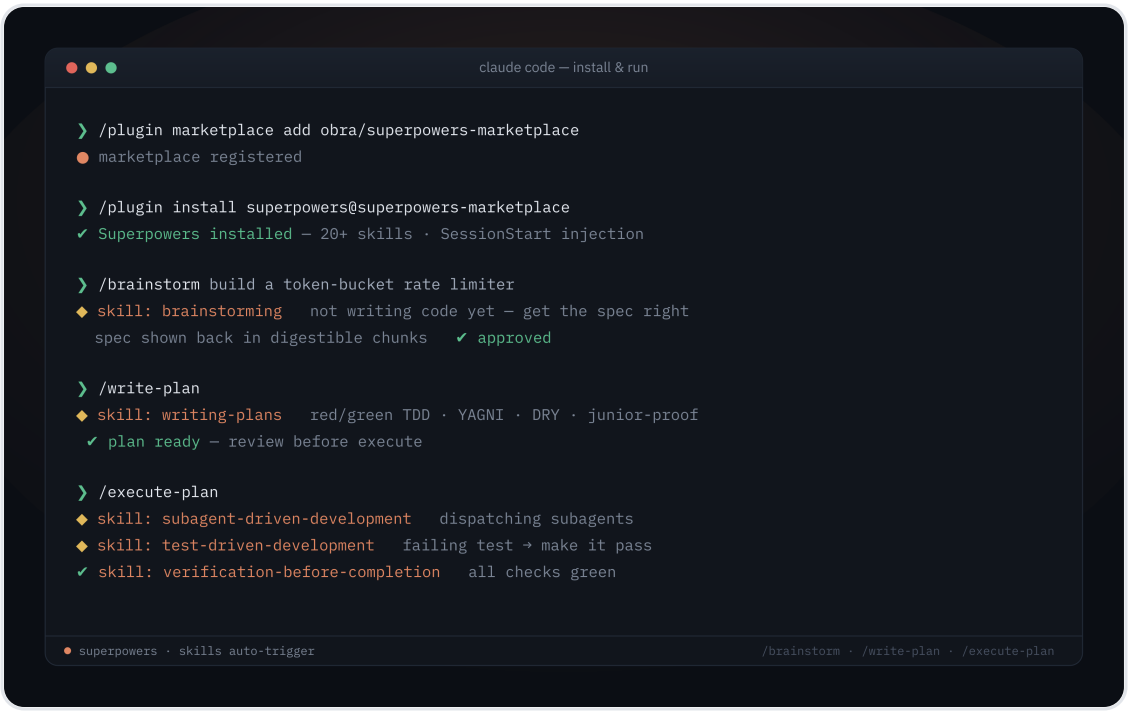
## How it works

1. **Understand**
   It doesn't jump to code. It steps back and asks what you're actually trying to build.
2. **Spec**
   It teases out a specification and shows it back in chunks short enough to actually read and sign off on.
3. **Plan**
   It writes an implementation plan clear enough for an enthusiastic junior with no context to follow — true red/green TDD, YAGNI and DRY.
4. **Build**
   Subagent-driven development works the plan task by task, inspecting as it goes — often autonomously for a couple of hours without drifting from the plan.
5. **Verify**
   It inspects, reviews and verifies its own work against the plan before marking anything complete.

## Install in Claude Code

```
$/plugin marketplace add obra/superpowers-marketplace
$/plugin install superpowers@superpowers-marketplace
```

Also available on the official Claude plugin marketplace — and for Codex, Gemini CLI, Cursor, Copilot CLI, Factory Droid and OpenCode.




Install, then plan and execute — skills trigger automatically

## The skills library

More than 20 battle-tested skills, auto-discovered and auto-triggered, with skills-search for discovery and context injected at session start. A sample of what ships:

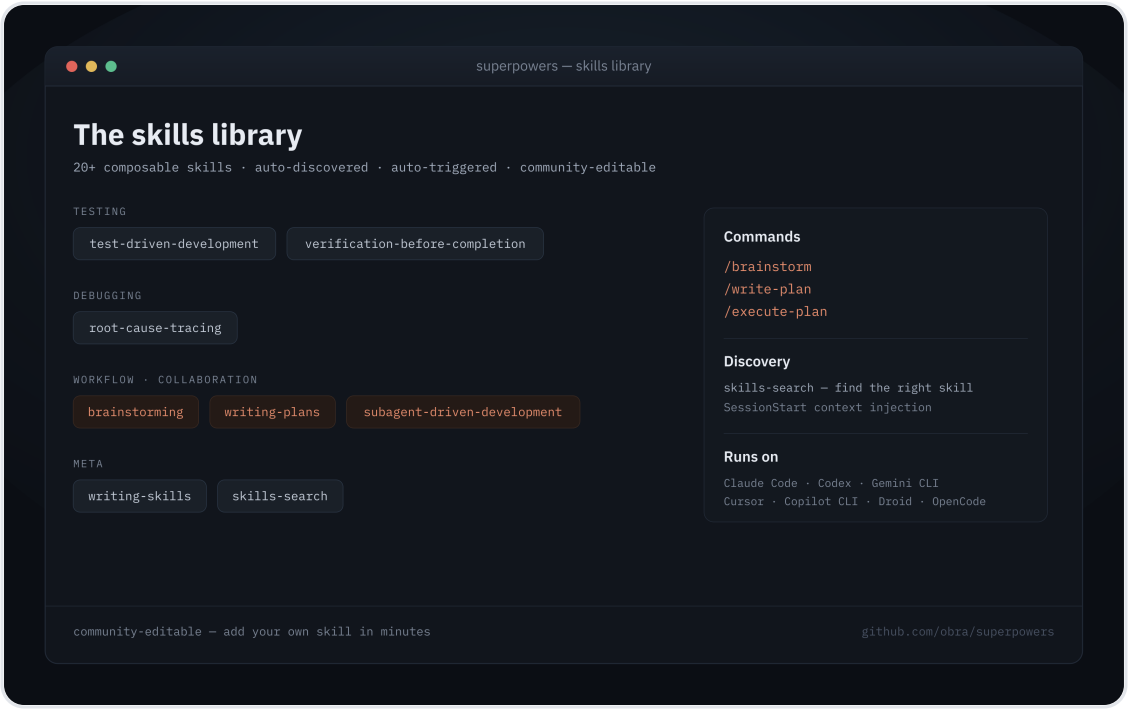
**Testing** `test-driven-development · verification-before-completion`

**Debugging** `root-cause-tracing`

**Workflow** `brainstorming · writing-plans · subagent-driven-development`

**Meta** `writing-skills · skills-search`

**Commands** `/brainstorm /write-plan /execute-plan`




20+ composable, community-editable skills

## Why it matters

The shift is simple but real: spec and plan before code, with a human signing off at each gate. It isn't slower — it's how you get autonomous output you can actually trust.