Skill Creator — Build Your Own Claude Skill

Anthropic's official meta-skill for creating, testing and optimizing skills.

BUILDER STACK

A **skill** is a folder with a `SKILL.md` file that gives Claude reusable, on-demand expertise for a task — a repeatable workflow instead of a one-off prompt. **Skill Creator** is Anthropic's official skill for building *other* skills the right way: it walks you from idea, to draft, to test prompts, to evaluation, to an optimized trigger description.

Who it's for

Anyone who finds themselves repeating the same instructions to Claude. If you've explained "how we write migrations" or "our PR checklist" more than twice, package it once as a skill and Claude reuses it forever.

Step 1. Install

The skill lives in Anthropic's public repo. Point Claude Code at it:

```
npx skills add anthropics/skills
```

Pick `skill-creator` when prompted (the repo also ships doc-coauthoring, docx, frontend-design and more). Or paste the repo link into Claude Code and say "install the skill-creator skill for me."

Step 2. Create a skill

Just describe what you want, e.g. "*Use skill-creator to make a skill for writing our API endpoint docs.*" Claude then runs the official loop:

1. **Capture intent** — it asks sharp questions (or reads the current conversation) to pin down exactly what the skill should do and how.
2. **Draft the SKILL.md** — a first version with a clear name and a precise `description` (the description is what makes Claude trigger the skill at the right time).
3. **Write test prompts** — a handful of realistic prompts to exercise the skill.
4. **Evaluate** — it runs the prompts against a fresh Claude that has the skill, then helps you review results qualitatively and with simple quantitative evals.
5. **Iterate** — rewrite based on what failed, then expand the test set and repeat.

> Prefer speed over rigor? Tell it "just vibe with me, skip the evals" and it will draft the skill without the full benchmark loop.

Step 3. Optimize triggering

Skills only help if Claude actually invokes them at the right moment — which depends almost entirely on the `description` field. Skill Creator ships a description-improver that rewrites your description for better trigger accuracy. Run it once the skill works, so it fires when you need it and stays quiet when you don't.

What makes a good skill

- **One job, done well.** Narrow skills trigger reliably; kitchen-sink skills confuse the model.
- **A precise description.** Say exactly *when* to use it ("Use when the user wants to…").
- **Concrete steps, not vibes.** Encode the real workflow, corrections and formats you use.
- **Test it.** The eval loop is the difference between a skill that works and one that looks like it does.