◆ FREE GUIDE

AI ESSENTIALS · GUIDE NO. 09

THE CLAUDE SKILLS PLAYBOOK

What a skill actually is, the 9 categories Anthropic uses internally, real examples, the best practices — and how to build your first one. Anthropic's own playbook, in plain English.

~10 MIN READ · FOR EVERYONE WHO COMMENTED "SKILLS" · CURRENT AS OF JULY 2026

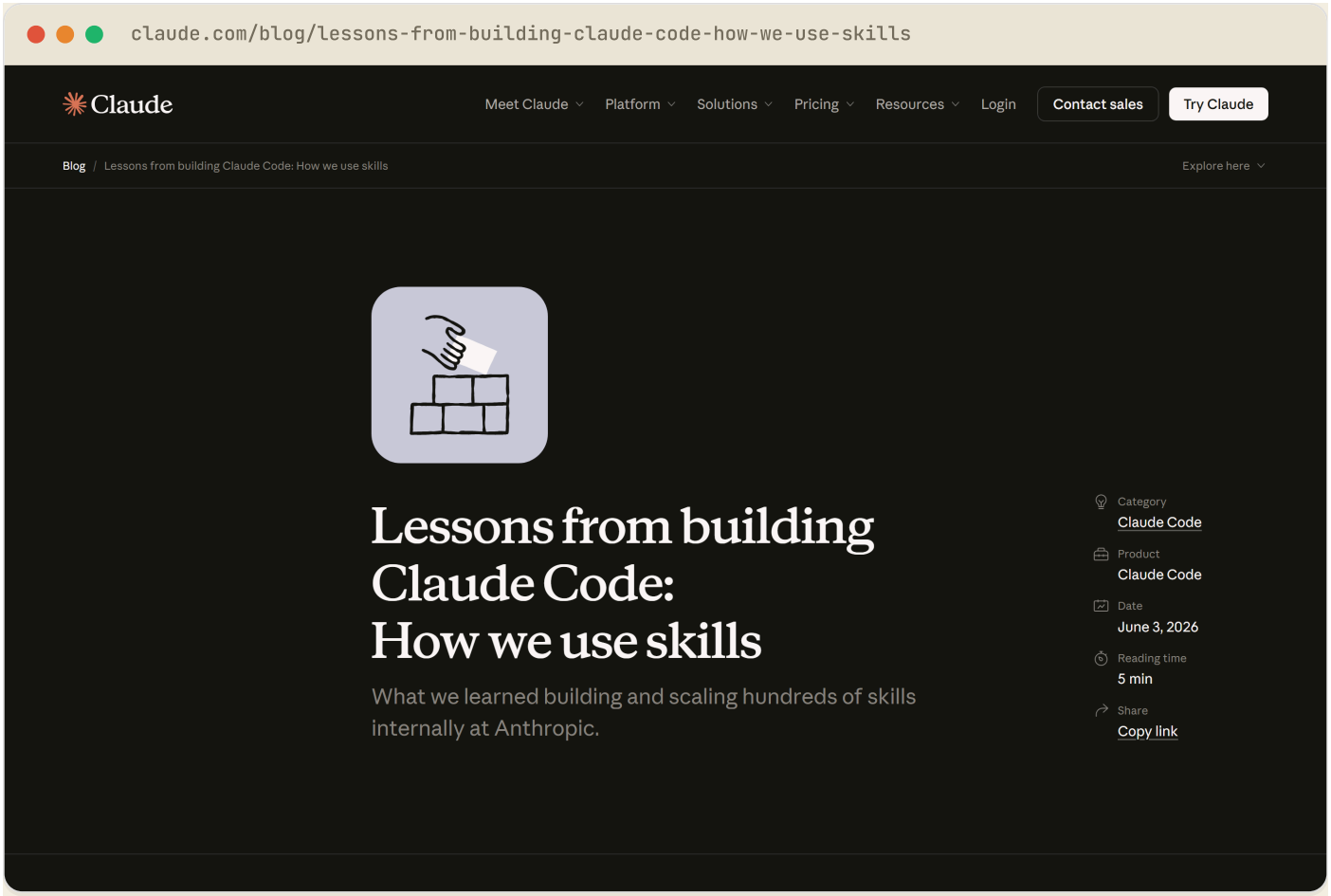
Hank Barker // AI Essentials // @hankhelpsai

THE SHORT VERSION

- → A **skill** is a folder that teaches Claude to do one task reliably — a `SKILL.md` file plus optional scripts, references, and templates.
- → The **description** is a trigger, not a summary. It's how Claude decides when to use the skill.
- → Anthropic's skills cluster into **9 categories** — and "product verification" (Claude testing its own work) had the biggest measurable impact.
- → You don't have to write any files — say **"turn this into a skill"** and Claude interviews you and builds it (via its `skill-creator` skill).

WHY THIS MATTERS

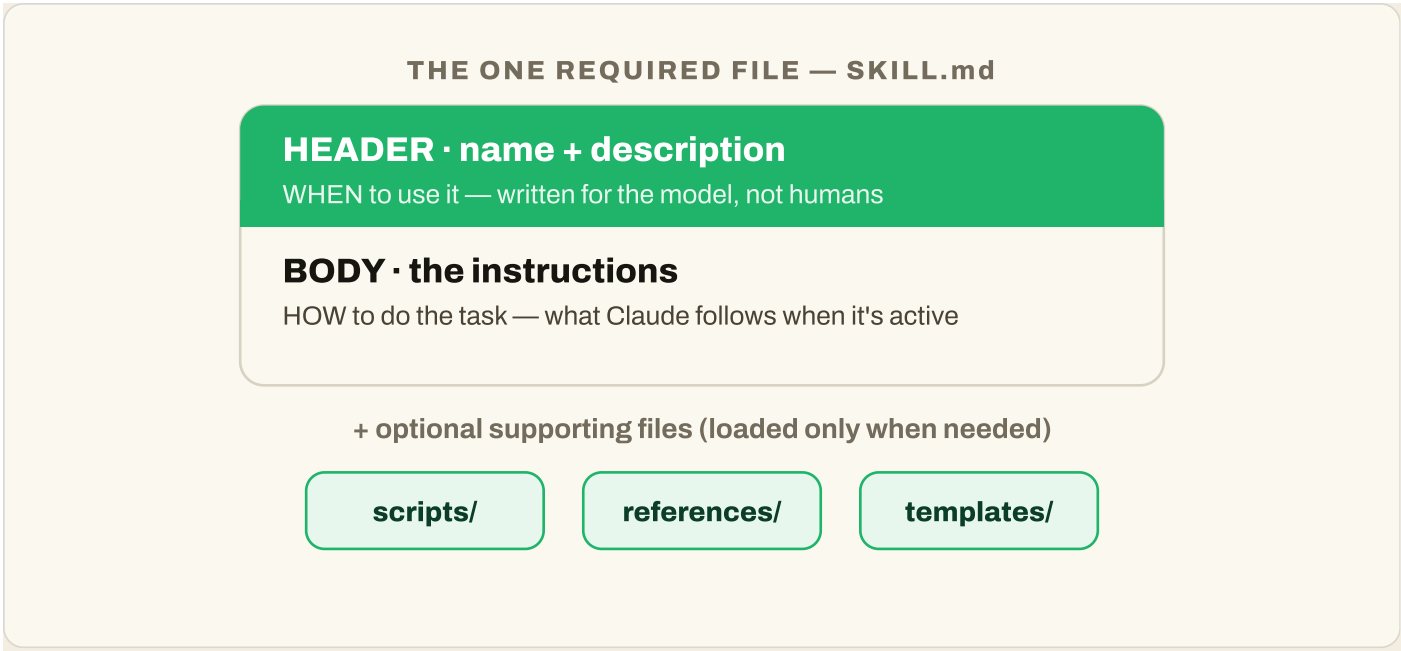
The people getting jaw-dropping, *consistent* results out of Claude aren't using a secret model. They've taught it their context. **Skills** are the cleanest way to do that: a reusable folder that turns "Claude sort of knows how to do this" into "Claude does this exactly the way we do, every time." Anthropic runs hundreds of them internally, and they published how — this guide is that playbook, translated.



Anthropic's write-up on how they build and scale skills internally — the source for this guide. — **Source:** Anthropic — "How we use skills"

WHAT A SKILL ACTUALLY IS

A skill is **a folder**, not a prompt. The one required file is `SKILL.md`, and it has two parts: a short **header** that tells Claude *when* to use the skill, and a **body** with the actual instructions. Everything else — scripts, reference docs, templates, data — is optional, and Claude only reads it when it needs to.



One required file, two parts, plus optional supporting files that load on demand.

WHERE SKILLS LIVE

TYPE	LOCATION	APPLIES TO
Global	`~/.claude/skills/`	Every project
Project	`.claude/skills/`	That one project
Plugin	Installed via the marketplace	Wherever you install it

Each skill is its own folder; at minimum it contains a `SKILL.md`.

THE SKILL.MD FORMAT

```
---
name: my-skill-name
description: When to trigger this skill — write it for the model.
---

# Instructions
Your instructions go here. Claude follows these when the skill is active.
```

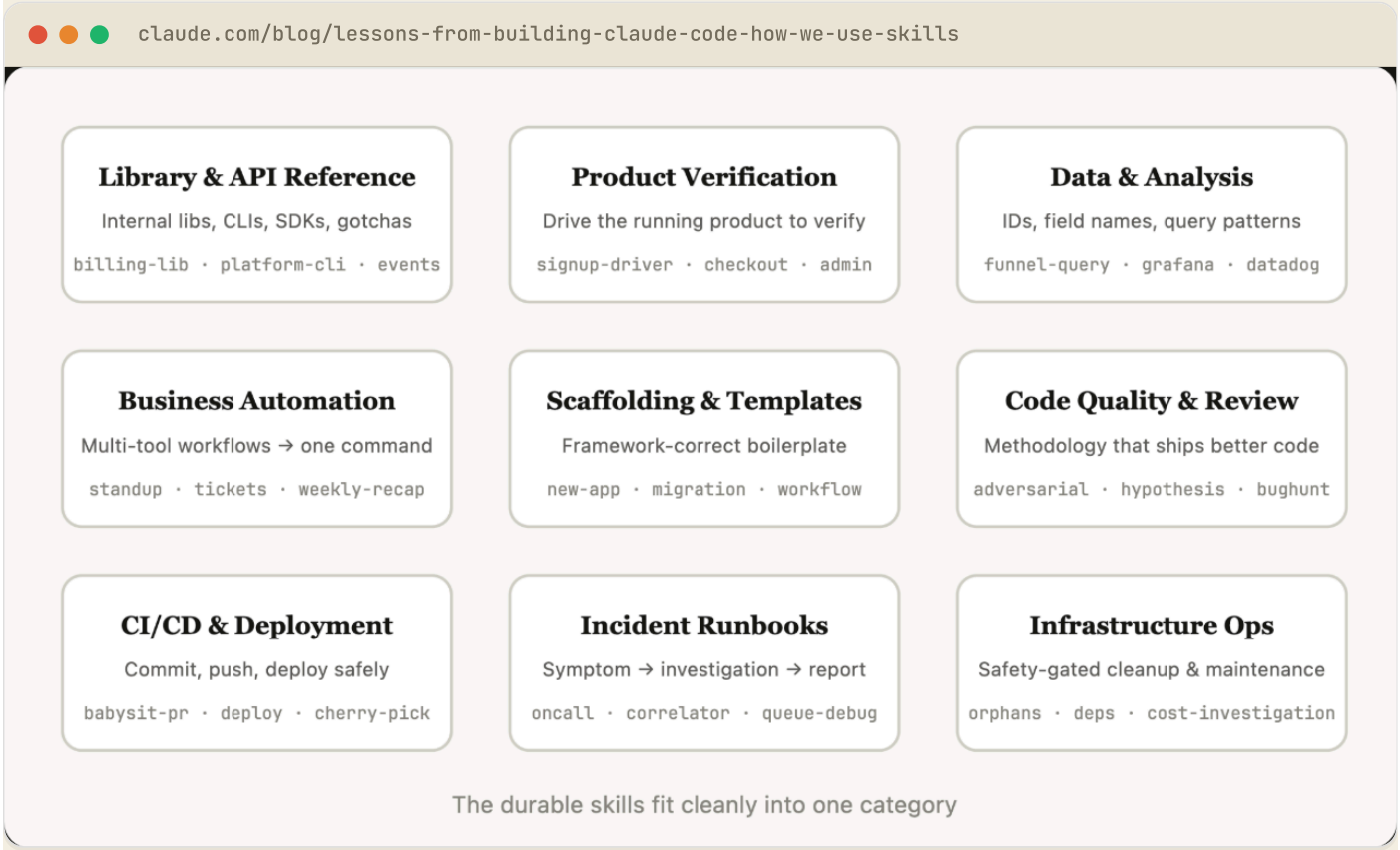
THE DESCRIPTION IS THE WHOLE GAME

When Claude starts a session it scans every skill's description to decide which are relevant. It's a **trigger condition**, not a summary — include the words a user would actually say.

✅ "Use when reviewing pull requests or checking code quality." ❌ "A helpful code review tool."

THE 9 CATEGORIES

After cataloguing all their internal skills, Anthropic found they fall into nine buckets. This is the map of what skills are *for*:



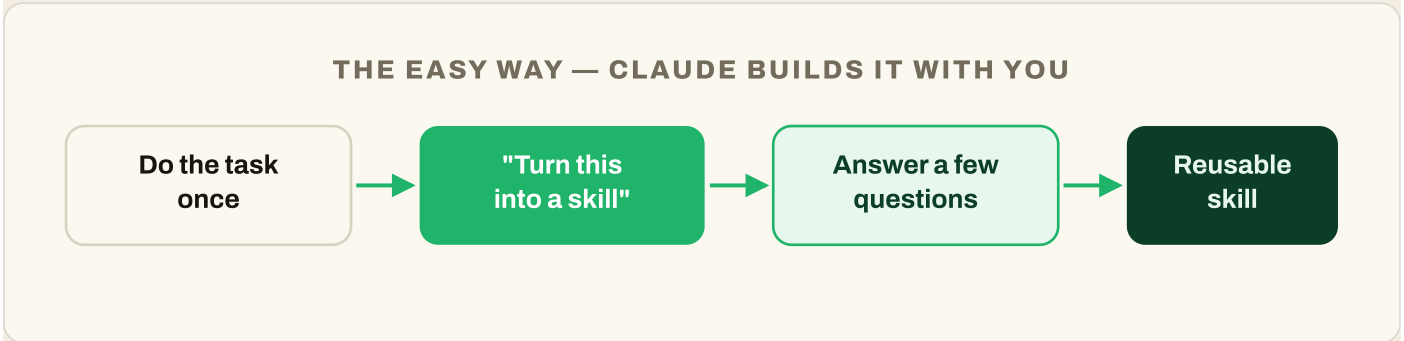
The nine categories, with real example skill names under each. — **Source:** Anthropic

- **1. Library & API reference** — correct usage + a *gotchas* section for tools Claude gets wrong (internal libs, freshly-changed APIs).
- **2. Product verification** — Claude tests its own work by driving the real product (signup flows, checkout with test cards, headless browser). **Anthropic says this had the biggest measurable impact on quality.**
- **3. Data fetching & analysis** — dashboard IDs, query patterns, and lookup tables so Claude pulls real numbers.
- **4. Business process automation** — repetitive workflows (standup, weekly recap, ticket creation) collapsed into one command.
- **5. Code scaffolding & templates** — boilerplate that follows *your* project's patterns, not generic defaults.
- **6. Code quality & review** — e.g. an adversarial review that spawns a fresh-eyes subagent to critique, then fixes.
- **7. CI/CD & deployment** — babysit a PR, roll out a deploy with error-rate checks, cherry-pick to prod safely.
- **8. Incident runbooks** — turn a symptom (alert, error, Slack thread) into a multi-tool investigation.
- **9. Infrastructure operations** — routine, often-destructive maintenance with guardrails and a soak period.

The highest-signal thing in any skill isn't the instructions — it's the gotchas. The failures Claude would otherwise repeat.

THE EASIEST WAY TO BUILD ONE: JUST ASK

Here's the part almost nobody realizes: you don't have to write a single file. Do a task once, then tell Claude **"turn this into a skill"** — and it builds the skill *with* you.



Do the work once, then let Claude package it into something reusable.

Claude uses its built-in `skill-creator` skill for this. It asks you a few clarifying questions — "when would you use this?", "what makes the output good?" — then writes the `SKILL.md`, organizes any files you handed it, generates any code it needs, and packages the whole thing. Skills 2.0 even runs a quick test on sample inputs before you approve it.

- **In Claude Cowork:** click the arrow next to the chat name → **"Turn into skill."**
- **On claude.ai:** say *"I want a skill that knows how to [X]"* or *"help me build a skill for [X]"* — and upload any templates or examples you'd want it to follow.
- **Then turn it on** in `Settings → Capabilities → Skills` and test it. You'll see "Using [skill name]" in Claude's thinking when it fires.

> **THE WHOLE LOOP**
>
> Do the task once → say "turn this into a skill" → answer a couple questions → it's reusable forever. No YAML, no folders, unless you want them.

WHAT ACTUALLY GOES INTO A GOOD SKILL

Whether Claude builds it or you do, the same ingredients separate a great skill from a mediocre one. This is the checklist — and the **description** and the **gotchas** do most of the heavy lifting.

- **A sharp description** — the trigger condition. The single most important line in the whole skill.
- **Clear steps** — what Claude should do, in order, including a verification step at the end.
- **A gotchas section** — the specific failures Claude would otherwise repeat. Highest-signal content there is.
- **Scripts** — for anything repeatable; a saved script beats re-deriving the code every time.
- **Reference docs** — detailed material Claude pulls in only when the task actually needs it.
- **Templates / assets** — the exact output format for Claude to copy.
- **A config file** — your specifics (which Slack channel, which repo) so the skill is reusable, not hard-coded.
- **One example of great output** — a single real example teaches more than a paragraph of description.

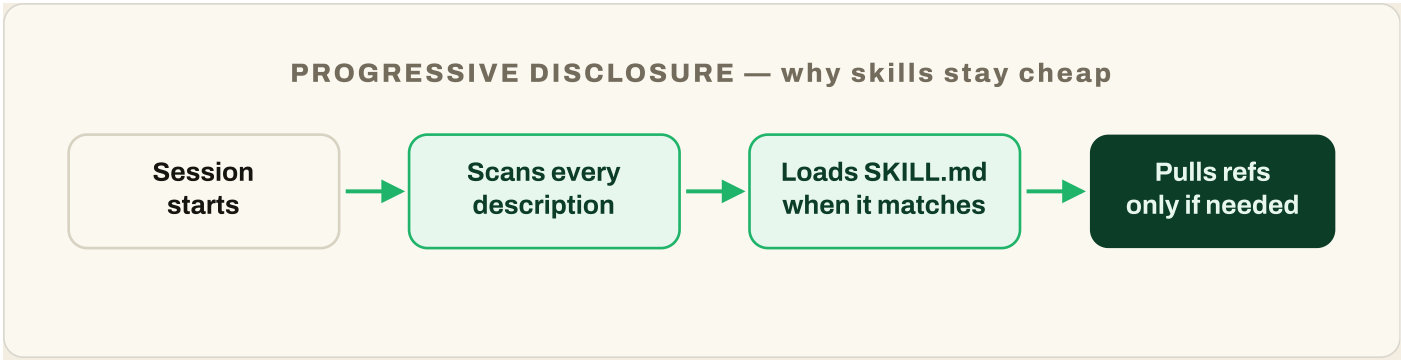
Under the hood it's still just a folder with a `SKILL.md` and optional supporting files. Here's the shape — whether Claude writes it or you do:

Only the SKILL.md is required. Everything else is optional and loads on demand.

```
---
name: my-skill
description: Use when [specific trigger condition]
---

# What this skill does
[One sentence.]

# Steps
1. [First thing Claude should do]
2. [Second thing]
3. [Verification step]

# Gotchas
- [Thing Claude will get wrong without this note]
- [Non-obvious constraint]
```

WHY SKILLS STAY CHEAP: PROGRESSIVE DISCLOSURE

The reason you can have a dozen skills without slowing Claude down: it doesn't read them all. It scans the short descriptions, loads the full `SKILL.md` only when one matches, and pulls the heavy reference files only if the task actually needs them. Keep the `SKILL.md` lean (Anthropic targets **1,500–2,000 words**) and push detail into `references/`.



Lean SKILL.md, heavy detail on demand — this is what keeps many skills fast.

MAKE IT REUSABLE WITH A CONFIG FILE

If a skill needs your specifics — which Slack channel, which repo — don't hard-code them. Store them in a `config.json` in the skill folder, and have the skill ask you the first time if it's missing. Here's exactly that pattern in a real Anthropic skill:

claude.com/blog/lessons-from-building-claude-code-how-we-use-skills

standup-post/SKILL.md

```
---
name: standup-post
description: Post your daily standup. Triggers on "standup", "daily".
---

## Your config
!`cat ${CLAUDE_SKILL_DIR}/config.json 2>/dev/null || echo "NOT_CONFIGURED"`

## Instructions
If the config above is NOT_CONFIGURED, ask the user:
- Which Slack channel?
- Paste a sample standup you liked
Then write the answers to ${CLAUDE_SKILL_DIR}/config.json.

Otherwise, post to the saved channel using the saved format.
```

The !`...` line runs as a shell command before Claude reads the prompt

The skill above is written to prompt the user if the Slack channel is not included in the configuration.

A standup skill that checks for config and interviews you if it's not set up yet — the "!" line runs as a shell command before Claude reads the prompt. — **Source:** Anthropic

BEST PRACTICES FROM ANTHROPIC

- **Don't state the obvious.** Claude can already code and read your repo. Spend the skill on what pushes it out of its default behaviour.
- **Build a gotchas section.** The single highest-value content. Grow it from real failures over time.
- **Keep SKILL.md lean** (1,500–2,000 words). Detail goes in `references/`.
- **Write descriptions for the model** — a trigger condition with the keywords users say, not a summary.
- **Give Claude scripts, not just instructions.** A script that fetches data beats re-deriving the code every time.
- **Don't overload.** 8–12 well-chosen skills cover most of a developer's day; more and you pay a context tax.

MEASURING & IMPROVING

Anthropic logs skill usage with a `PreToolUse` hook so they can see which skills are popular and which under-trigger. The pattern to copy: the best skills start as a few lines and a single gotcha, then get sharper every time Claude hits a new edge case and you add what it missed.

WHAT TO DO NEXT

Pick one task you explain to Claude the same way every week. Do it once, then say **"turn this into a skill"** and answer its questions. Tomorrow it's one command — and that first "oh, it just did it right" is the whole point.

SOURCES & WHERE TO LEARN MORE

- ↗ Anthropic — Lessons from building Claude Code: How we use skills
- ↗ Claude — How to create a skill through conversation (official tutorial)
- ↗ Claude Docs — Agent Skills (how to create & install)
- ↗ Want the connectors playbook too? Comment "SYSTEM" for that guide

Built by Hank Barker // AI Essentials // @hankhelpsai. Distilled from Anthropic's own skills write-up, July 2026.