Find, Check, and Install Your First Claude Skill

Most people run Claude with its whole "skills" layer switched off — and the ones who turn it on install blindly. Here's the safe way to do both, in about ten minutes.

Here's the thing almost nobody tells you: out of the box, your agent knows a lot in general and nothing about your specific jobs. A "skill" is how you hand it a specific job done right — and there are now tens of thousands of them sitting in the open, free to grab. Most people never touch that layer. The few who do usually paste in whatever they find without checking it, which is its own problem because these things run real code on your machine.

This guide fixes both. You'll learn where the skills live, how to see which ones are safe *before* you install, and the exact three commands to get your first one working today.

What you're missing right now

A raw agent is a smart generalist. Ask it to do a real, specific task — write a Reel script in your voice, scrape a page, review a PR the way your team reviews PRs — and it improvises from general knowledge. Sometimes fine. Often almost-right in a way that costs you more time to fix than to have done yourself.

A **skill** is a folder that teaches your agent one job the right way — the steps, the rules, the format, the gotchas — and it loads automatically the moment that job comes up. You don't prompt harder. The agent just already knows how you want it done.

The catch: the folks who *do* discover skills tend to grab the first one they find and install it without a second look. And a skill isn't a passive document — it can run shell commands, read your files, and touch your API keys. There have been real cases of malicious skills in the wild. So "just install it" isn't the move. "Check it, then install it" is. That's the whole game, and it takes one extra step.

What it feels like when it clicks

You ask your agent to do a thing. Instead of a generic best-guess, it quietly pulls in the right skill and does the job the way a specialist would — right steps, right format, the small stuff handled — without you spelling any of it out.

You stop writing three-paragraph prompts to get one job done right. You install the skill once, and every time that job comes up, it's handled. That's the difference between a chatbot you keep correcting and a tool that already knows the work.

The 3-step play

Where all of this lives

skills.sh — *The Agent Skills Directory*, built by Vercel Labs. It indexes tens of thousands of open agent skills — coding, research, writing, scraping, all of it — and it works across agents (Claude Code, Cursor, Codex). This is your shelf. Start here.

> **What a skill actually is:** a `SKILL.md` file (plus any helper files) that teaches your agent one job. In Claude Code it lives at `.claude/skills/<name>/SKILL.md` and loads on its own when it's relevant.

Step 1 — FIND the skill

Two ways. Pick either.

A. Let your agent find it for you. There's a skill whose only job is finding other skills — it's the most-installed skill on the directory, with millions of installs. You install it once, then just tell your agent what you're trying to do and it finds and installs the right skill for the task.

```
npx skills add vercel-labs/skills
```

That's **find-skills**. After it's in, you describe the job in plain words and it handles the search.

B. Search the shelf yourself — from your terminal:

```
npx skills find "web scraping"
```

...or just browse **skills.sh** and read what's there. Same result, your eyes on it.

Step 2 — CHECK the badges (don't skip this)

This is the step that keeps you safe, and it's the one everyone skips.

Every skill on the directory carries **third-party audit badges** — see them all at **skills.sh/audits**. You get to see whether something is clean or sketchy *before* it ever touches your machine:

- **Gen Agent Trust Hub** — a plain **Safe** or **High-Risk** call.
- **Socket** — a count of security alerts found in the code.

- **Snyk** — a severity rating: **Low / Medium / High / Critical**.

Read it like a traffic light. Green across the board, install it. Alerts or a high-risk flag, slow down and look closer — or pick a cleaner alternative doing the same job.

One more 20-second habit: **open the `SKILL.md` and actually read it** before you install. It's plain text. You're checking what it says it does — and, if you can, what commands it runs. Badge plus a quick read of the file is 95% of your safety, for basically no effort.

Step 3 — INSTALL it (one command)

Once it's passed the check:

```
npx skills add <owner/repo>
```

Point it at a specific agent with `-a`:

```
npx skills add <owner/repo> -a claude-code
```

See everything you've installed:

```
npx skills list
```

That's it. It's in, it'll load when it's relevant, and you never had to write a prompt to explain the job.

The official route (if you'd rather stay first-party)

Claude Code has its own plugin system, straight from Anthropic. Add the official marketplace, then install from it:

```
/plugin marketplace add anthropics/claude-plugins-official
/plugin
```

Same idea, first-party source. Either way, your skills end up living at `.claude/skills/<name>/SKILL.md`.

The bigger picture: skills are one piece

A skill on its own is already worth it. But skills are one of three things you can bolt onto your agent, and they're better together:

- **Skills** — teach it *how* to do specific jobs (the part we just covered).
- **Subagents** — specialized helpers that work in their own context, so your main thread stays clean. A researcher, a reviewer, a test-writer running in parallel.
- **Hooks** — your own commands that fire automatically on events. Auto-format after every edit. Block edits to a protected file. Rules that enforce themselves.

Stack them and the agent stops feeling like a chat box and starts working like it already knows your setup: subagents split the work, skills handle the repeatable jobs, hooks keep it inside your rules. You build it once. It runs every time after that.

Your next step

Do the small version today, not the big one. Open **skills.sh**, install `find-skills` with the one command above, check the badges on the first real skill you want, and install it. One job, done the right way, from now on. Then add the next one when you hit the next repeatable task.

Bookmark **skills.sh** and **skills.sh/audits** — the find and the check. That's the whole loop.

Built by Axionox — I build AI systems and automations for a living, and this is the setup I actually run every day. Want more like this? Follow along.