A TEAM GUIDE

Creating skills for Claude

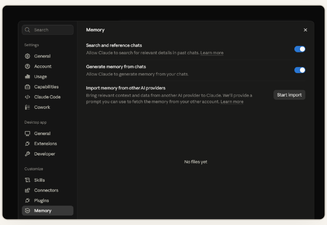
A skill packages a workflow you repeat into a folder Claude loads on its own when a task matches it. Write it once, and reuse it across Claude.ai, Claude Desktop, and Claude Code.

START HERE

Not sure where to begin?

You don't need a plan. Let Claude find your skill candidates for you.

1. **Turn on chat memory**
   Settings → Customize → Memory, and switch on "Search and reference chats." This lets Claude look back through your past chats.
2. **Open a project, start a new chat**
   Keep related work in a project so Claude can scan it as a set.
3. **Paste the starter prompt and let it work**
4. **Review what it finds**
   Pick one, build it, and click **Save skill**.



Step 1: both memory toggles on, under Customize → Memory.

COPY-PASTE STARTER PROMPT

"Look through the chats in this project and find repetitive things I do that could become skills."

1 What a skill actually is

- A folder with one file, **SKILL.md**: YAML frontmatter (a name and a description) followed by Markdown instructions.
- The same format works everywhere — the open "Agent Skills" standard, so one folder runs in Claude.ai, Claude Desktop, and Claude Code unchanged.
- Claude loads a skill only when a task matches its description, so it costs nothing until it's needed.

2 When to make one

Watch for work you keep repeating in chat:

- A prompt you paste again and again.
- A document you keep rebuilding to the same shape.
- An intake or review you always run the same way.

> **Rule of thumb: if you've done it three or more times, it's a skill candidate.**

3 The description is the trigger

- Two required frontmatter fields: **name** and **description**.
- The description is what makes Claude reach for the skill. Make it assertive about both what it does and when to use it. Vague descriptions mean the skill never fires, so lean slightly "pushy."
- Keep the body under ~500 lines; push heavy detail into separate reference files the skill points to.

4 How to write a good one

- **Capture the pattern from real examples.** Pull the recurring structure out of past work — don't invent it from scratch.
- **Encode the judgment, not just the steps.** Write down the rules that make your version good.
- **Use the imperative and explain why** each rule matters — Claude follows reasons better than bare commands.
- **Add a worked example** so the shape is unambiguous.
- **Sanity-check it.** Run it against a real past case, compare with what you built by hand, then adjust.

5 How to install — easiest first

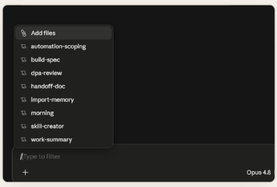
START HERE **Build it in a chat, then click Save**

Ask Claude to create the skill in a normal conversation or project. When it presents the SKILL.md file, click **"Save skill"** on the file card and it saves into your profile automatically — no folders, no uploads, no settings to dig through.

Claude Code & Claude Desktop — drop the folder in place

Personal, available in every project:
`~/.claude/skills/<name>/SKILL.md`
Project, travels with a repo:
`.claude/skills/<name>/SKILL.md`
Add `user-invocable: true` for a `/<name>` slash command. Skills you save on Claude.ai can also sync down to Claude Code.

Manual upload (alternative): Settings → Capabilities → Skills.



Installed skills show up when you type / in the chat box.

6 A real (trimmed) example

The top of our chat-recap skill. Frontmatter first, then the instructions.

```
---
name: work-summary
description: >-
  Produce a tight 4-6 sentence recap of the current work session,
  focused on practical work and outcomes. Use whenever the user asks
  to summarize, recap, wrap up, or "log" a conversation or project
  chat. Always cover the goal, the concrete tasks, the effort/friction,
  and the deliverable, and call out coding or repetitive work explicitly.
user-invocable: true   # Claude Code: also creates /work-summary
argument-hint: "[audience: self | manager | leadership | handoff]"
---
# Work Summary

Recap a work session in a fixed, scannable shape...

## Output format

Write 4-6 sentences of prose (not bullets) covering, in order:
1. What the user was trying to accomplish
2. The concrete tasks involved
3. The manual effort or friction / where it got stuck
4. The end result or deliverable
```

The Claude-Code-only fields (`user-invocable`, `argument-hint`) are harmlessly ignored on Claude.ai — so one file covers every surface.

Done it three times? Make it a skill.

sharicewells.com