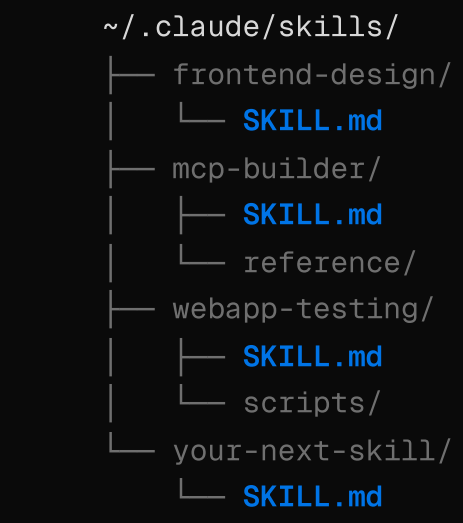
Idlen



The best Claude Skills

15 skills worth installing for building, designing, documenting and testing with Claude and Claude Code. With install commands, and how to write your own.

---

by Idlen

For vibe coders, indie hackers, founders and devs.
Every skill and link checked on 31 October 2026.

What a skill is

A skill is a folder with a `SKILL.md` file in it. That's the whole format.

The file starts with a short YAML header, a `name` and a `description`, followed by plain Markdown instructions. The folder can also hold scripts, templates and reference docs that Claude opens only when the instructions point to them.

Claude doesn't read every skill up front. It loads in three steps:

1. **Name and description** are always loaded, about 100 tokens per skill according to Anthropic's docs.
2. **The body of SKILL.md** is read only when your request matches the description.
3. **Bundled files** are read or run only when needed. When Claude runs a script, only its output enters the context.

So you can install a dozen skills without paying for them in every message. And the description is what decides whether a skill fires, which matters when you write your own (page 9).

Where skills work

Claude.ai	Anthropic's document skills (Word, Excel, PowerPoint, PDF) run when file creation is on. You can upload your own as a ZIP.
Claude Code	Skills are folders on disk: `~/.claude/skills/` for all projects, `.claude/skills/` for one repo. Plugins can ship skills too.
Claude API	Pass skill IDs in the `container` parameter alongside the code execution tool.

Skills follow an open format, Agent Skills (agentskills.io), so the same folder can be reused across surfaces. Uploads don't sync between claude.ai and the API, but Claude Code signed in with your claude.ai account loads the skills you enabled there.

> **Read before you install.** A skill can tell Claude to run code with the same access Claude has on your machine. Anthropic recommends using skills only from sources you trust and reviewing the files, scripts included, of anything else.

Anatomy of a skill folder

`mcp-builder/`	The folder. Its name should match the skill's `name`.
`SKILL.md`	**Required.** YAML header (always loaded) plus instructions (loaded when the skill fires).
`reference/`	Optional docs Claude reads only when SKILL.md tells it to.
`scripts/`	Optional code Claude runs. The code itself stays out of the context; only the output comes back.

This guide covers 15 skills in five groups, from three sources: Anthropic's official repository, Jesse Vincent's *Superpowers* collection and Vercel's agent skills. Page 10 has all of them on one page.

Quick start

Three ways in. Pick the one that matches where you work.

Claude Code

A Install Anthropic's skills as plugins. Type this inside a Claude Code session:

```
> /plugin marketplace add anthropics/skills
> /plugin install example-skills@anthropic-agent-skills
> /plugin install document-skills@anthropic-agent-skills
```

B Install one skill at a time from any GitHub repo with the open-source `skills` CLI (made by Vercel). Run it in your terminal:

```
$ npx skills add anthropics/skills --skill mcp-builder -a claude-code
```

It copies the folder to `./.claude/skills/` in the current project. Add `-g` to install it for all your projects.

C By hand: put the folder at `~/.claude/skills/<name>/SKILL.md` (all projects) or `.claude/skills/<name>/SKILL.md` (one repo). Claude Code picks up changes without a restart.

> **Using a skill.** Describe the task and Claude loads the skill when the description matches, or type `/skill-name` to run it directly. `/skills` lists what's loaded.

Claude.ai

1. Settings › Capabilities: turn on *Code execution and file creation*.
2. Customize › Skills: switch Anthropic's skills on or off.
3. Your own skill: ZIP the folder, then **+** › Create skill › Upload a skill.

The folder name has to match the skill's `name`, or the upload fails. The document skills (Word, Excel, PowerPoint, PDF) are used automatically once file creation is on.

Claude API

```
client.messages.create(
    model="claude-opus-5-5",
    max_tokens=4096,
    container={"skills": [{"type": "anthropic",
        "skill_id": "xlsx", "version": "latest"}]},
    tools=[{"type": "code_execution_20250825",
            "name": "code_execution"}],
    messages=[...])
```

Pre-built IDs: `pptx`, `xlsx`, `docx`, `pdf`. Upload your own through the `/v1/skills` endpoints. Skills on the API run in a sandbox with no network access.

Docs: code.claude.com/docs/en/skills · support.claude.com/en/articles/12512180 · platform.claude.com/docs/en/build-with-claude/skills-guide

Build & ship

For when the thing you're building talks to Claude, or other tools talk to it.

claude-api

Anthropic

Reference for the Claude API and Anthropic SDKs: model IDs, parameters, streaming, tool use, prompt caching, token counting, and migrating code to newer models.

Use it when: You're building an app that calls Claude and want Claude to use the current API, not what it remembers from training.

Install:

```
Bundled in Claude Code: /claude-api  (e.g. /claude-api migrate)
```

```
npx skills add anthropics/skills --skill claude-api -a claude-code
```

Source: github.com/anthropics/skills/tree/main/skills/claude-api

mcp-builder

Anthropic

Guide for building MCP servers in Python (FastMCP) or TypeScript (MCP SDK), in four phases: research and plan, implement, review and test, then write evaluations that check an LLM can actually use the tools.

Use it when: You want Claude, or any MCP client, to work with your product or an external API through well-designed tools.

Install:

```
/plugin install example-skills@anthropic-agent-skills
```

```
npx skills add anthropics/skills --skill mcp-builder -a claude-code
```

Source: github.com/anthropics/skills/tree/main/skills/mcp-builder

web-artifacts-builder

Anthropic

Scripts that set up a React 18, TypeScript, Vite, Tailwind and shadcn/ui project, then bundle it into a single HTML file that runs as a claude.ai artifact.

Use it when: Your prototype or internal tool needs state, routing or many components. For a simple one-file artifact, skip it.

Install:

```
/plugin install example-skills@anthropic-agent-skills
```

```
npx skills add anthropics/skills --skill web-artifacts-builder -a claude-code
```

Source: github.com/anthropics/skills/tree/main/skills/web-artifacts-builder

writing-plans

obra/superpowers

Turns a spec into a step-by-step implementation plan written for an engineer who has never seen the codebase: which files to create or change and the exact test for each step. Pairs with `brainstorming`, which turns a rough idea into that spec first.

Use it when: Before a feature that will take more than a few edits, so Claude Code works from a plan instead of improvising.

Install:

```
/plugin install superpowers@claude-plugins-official  (whole set)
```

```
npx skills add obra/superpowers --skill writing-plans -a claude-code
```

Source: github.com/obra/superpowers/tree/main/skills/writing-plans

Design & front-end

For UIs that don't look like every other generated landing page, and that hold up in review.

---

frontend-design

Anthropic

Design direction for new UI or a redesign: picking an aesthetic, typography and layout on purpose. It lists the traits that make generated designs look alike so Claude can steer away from them.

Use it when: You're starting a landing page, dashboard or app screen and the first draft looks templated.

Install:

```
/plugin install frontend-design@claude-plugins-official
```

```
npx skills add anthropics/skills --skill frontend-design -a claude-code
```

Source: github.com/anthropics/skills/tree/main/skills/frontend-design

web-design-guidelines

Vercel

Reviews UI code against Vercel's Web Interface Guidelines. It fetches the latest version of the rules on each run and reports findings as `file:line`.

Use it when: You ask "review my UI", "check accessibility" or "audit the UX" before shipping.

Install:

```
npx skills add vercel-labs/agent-skills --skill web-design-guidelines -a claude-code
```

Source: github.com/vercel-labs/agent-skills/tree/main/skills/web-design-guidelines

vercel-react-best-practices

Vercel

React and Next.js performance rules from Vercel Engineering, in 8 categories ordered by impact. Request waterfalls and bundle size come first, then server-side performance, data fetching and re-renders.

Use it when: Writing, reviewing or refactoring React or Next.js code, or chasing a slow page.

Install:

```
npx skills add vercel-labs/agent-skills --skill vercel-react-best-practices -a claude-code
```

Source: github.com/vercel-labs/agent-skills/tree/main/skills/react-best-practices

Docs & office files

The skills behind Claude's file creation. On in Claude.ai once file creation is enabled, available in the API by ID, and installable in Claude Code as one plugin.

---

pdf

Anthropic

Read and extract text and tables, merge, split, rotate, watermark, create new PDFs, fill forms, encrypt or decrypt, extract images, and OCR scanned pages.

Use it when	Invoices, contracts, forms, reports. Anything where a PDF goes in or comes out.
Install	`/plugin install document-skills@anthropic-agent-skills`
	`API: skill_id "pdf"`
Source	github.com/anthropics/skills/tree/main/skills/pdf

docx

Anthropic

Create and edit Word documents and templates: tables of contents, headings, page numbers, letterheads, images, find-and-replace, tracked changes and comments.

Use it when	You need a real .docx: a proposal, a memo, a contract draft a client will edit in Word.
Install	`/plugin install document-skills@anthropic-agent-skills`
	`API: skill_id "docx"`
Source	github.com/anthropics/skills/tree/main/skills/docx

pptx

Anthropic

Create, read, edit, combine and split slide decks, including templates, layouts and speaker notes.

Use it when	A pitch deck, an investor update, or pulling the text out of someone else's deck.
Install	`/plugin install document-skills@anthropic-agent-skills`
	`API: skill_id "pptx"`
Source	github.com/anthropics/skills/tree/main/skills/pptx

xlsx

Anthropic

Open, fix and create spreadsheets (.xlsx, .xlsm, .csv, .tsv): formulas, formatting, charts, cleaning messy data, converting between formats.

Use it when	A financial model, a cleaned-up export, or a report your accountant can open.
Install	`/plugin install document-skills@anthropic-agent-skills`
	`API: skill_id "xlsx"`
Source	github.com/anthropics/skills/tree/main/skills/xlsx

These four are source-available rather than open source. The repository shares them as a reference for how a production skill is built.

Testing & debugging

For checking that what Claude built works, and fixing it properly when it doesn't.

webapp-testing

Anthropic

Tests local web apps with Python Playwright scripts. A helper script starts your dev server (or several) and shuts it down after. Claude can click through flows, take screenshots and read browser logs.

Use it when: Claude just changed a UI flow and you want it to check in a real browser instead of saying "should work".

Install:

```
/plugin install example-skills@anthropic-agent-skills
```

```
npx skills add anthropics/skills --skill webapp-testing -a claude-code
```

Source: github.com/anthropics/skills/tree/main/skills/webapp-testing

systematic-debugging

obra/superpowers

Root cause first, fix second. Four phases: investigate the root cause, look for the pattern, test one hypothesis at a time, then implement. It also lists the signs that a debugging session has gone off track.

Use it when: Any bug or failing test, and especially after two quick fixes that didn't stick.

Install:

```
/plugin install superpowers@claude-plugins-official
```

```
npx skills add obra/superpowers --skill systematic-debugging -a claude-code
```

Source: github.com/obra/superpowers/tree/main/skills/systematic-debugging

test-driven-development

obra/superpowers

Red, green, refactor: write a failing test, watch it fail, write the minimum code to pass, then clean up. No production code before a failing test.

Use it when: Features and bug fixes where you want proof the code works, not just code that looks right.

Install:

```
/plugin install superpowers@claude-plugins-official
```

```
npx skills add obra/superpowers --skill test-driven-development -a claude-code
```

Source: github.com/obra/superpowers/tree/main/skills/test-driven-development

Creating your own

The best skill is often the one that writes down how you already work.

`skill-creator`

Anthropic

Helps you draft a new skill or improve an existing one, then measure it: it runs test prompts with and without the skill, grades the outputs, benchmarks pass rate, time and tokens, and suggests edits to the description so the skill triggers on the right requests.

Use it when: You keep pasting the same instructions into chat, or a skill you wrote fires at the wrong times.

Install:

```
/plugin install skill-creator@claude-plugins-official
```

```
npx skills add anthropics/skills --skill skill-creator -a claude-code
```

Source: github.com/anthropics/skills/tree/main/skills/skill-creator

---

Good first skills to write

Your stack

How your repo is laid out, the commands to run tests and migrations, the libraries you use and the ones you don't.

Your deploy

The exact release steps. In Claude Code, set `disable-model-invocation: true` so it only runs when you type it.

Your voice

How you write changelogs, launch posts and support replies, with two or three real examples.

Your checklist

What you check before merging: accessibility, error states, analytics events, copy.

Templates to start from: the `template` folder in github.com/anthropics/skills, or run `npx skills init my-skill`.

Write your own in ten minutes

One folder, one file. This one writes release notes from your git history.

Save as `~/.claude/skills/release-notes/SKILL.md`

```
---
name: release-notes
description: Writes release notes from the git log since the
  last tag. Use when the user asks for release notes, a changelog
  entry, or what shipped.
---

# Release notes

1. Run `git describe --tags --abbrev=0` to find the last tag.
2. Run `git log <tag>..HEAD --oneline` and group the commits
   into Added, Fixed and Changed.
3. Write one plain sentence per change, for users, not developers.
   Skip chores and merge commits.
4. Output Markdown. Never list a change that isn't in the log.
```

Rules that matter

- **name**: lowercase letters, numbers and hyphens, 64 characters max, no "anthropic" or "claude".
- **description**: up to 1,024 characters. Say what it does *and* when to use it. Claude matches your request against this text, so use the words you'd actually type.
- Keep `SKILL.md` under 500 lines. Move long reference material into separate files and link to them.
- For claude.ai or the API, stick to the standard fields: `name`, `description`, `license`, `compatibility`, `metadata`, `allowed-tools`.

Test it

- Open a fresh session and ask something that should trigger it: "what shipped since the last release?"
- Or call it directly: `/release-notes`.
- If it doesn't fire, rewrite the description with the words you used. If it fires too often, make the description narrower.
- To share it with your team, commit `.claude/skills/` to the repo.

Rather not write it yourself? Install `skill-creator` (page 8) and tell Claude what you keep repeating: "make a skill from how we write release notes". It drafts the folder, tests it on a few prompts and tunes the description.

All 15 on one page

Plugin names go after `/plugin install` in Claude Code. Any skill can also be installed with `npx skills add <repo> --skill <name> -a claude-code`.

Group	Skill	From	Fastest install in Claude Code
Build & ship	`claude-api`	Anthropic	`/claude-api (bundled in Claude Code)`
	`mcp-builder`	Anthropic	`example-skills@anthropic-agent-skills`
	`web-artifacts-builder`	Anthropic	`example-skills@anthropic-agent-skills`
	`writing-plans`	obra/superpowers	`superpowers@claude-plugins-official`
Design	`frontend-design`	Anthropic	`frontend-design@claude-plugins-official`
	`web-design-guidelines`	Vercel	`npx skills add vercel-labs/agent-skills`
	`vercel-react-best-practices`	Vercel	`npx skills add vercel-labs/agent-skills`
Docs & files	`pdf · docx · pptx · xlsx`	Anthropic	`document-skills@anthropic-agent-skills`
Testing	`webapp-testing`	Anthropic	`example-skills@anthropic-agent-skills`
	`systematic-debugging`	obra/superpowers	`superpowers@claude-plugins-official`
	`test-driven-development`	obra/superpowers	`superpowers@claude-plugins-official`
Meta	`skill-creator`	Anthropic	`skill-creator@claude-plugins-official`

Before the plugin commands work

```
> /plugin marketplace add anthropics/skills                 # for @anthropic-agent-skills
> /plugin marketplace add anthropics/claude-plugins-official  # only if Claude Code says it's not found
```

Sources

github.com/anthropics/skills

github.com/obra/superpowers

github.com/vercel-labs/agent-skills

github.com/vercel-labs/skills (npx skills CLI)

code.claude.com/docs/en/skills

platform.claude.com/docs › Agent Skills

support.claude.com › Use skills in Claude

agentskills.io (the open standard)

Idlen

Built something with AI? Advertise it where AI users wait.

Idlen shows native ads in the seconds people spend waiting for an AI to answer: in Chrome, Firefox and VS Code extensions, and in apps through the Idlen Chat SDK. The people seeing them use AI tools every day, which is who you built your product for.

Put in €50, get €100.

idlen.io/advertisers

This guide is free to share. Follow **@idlenads** on TikTok and **@idlen_ai** on Instagram for more on building with AI.