CLAUDE SKILLS · GUIDE 2 OF 3

Claude Skills Under the hood

How a skill is built, how Claude loads it, and where it's kept.

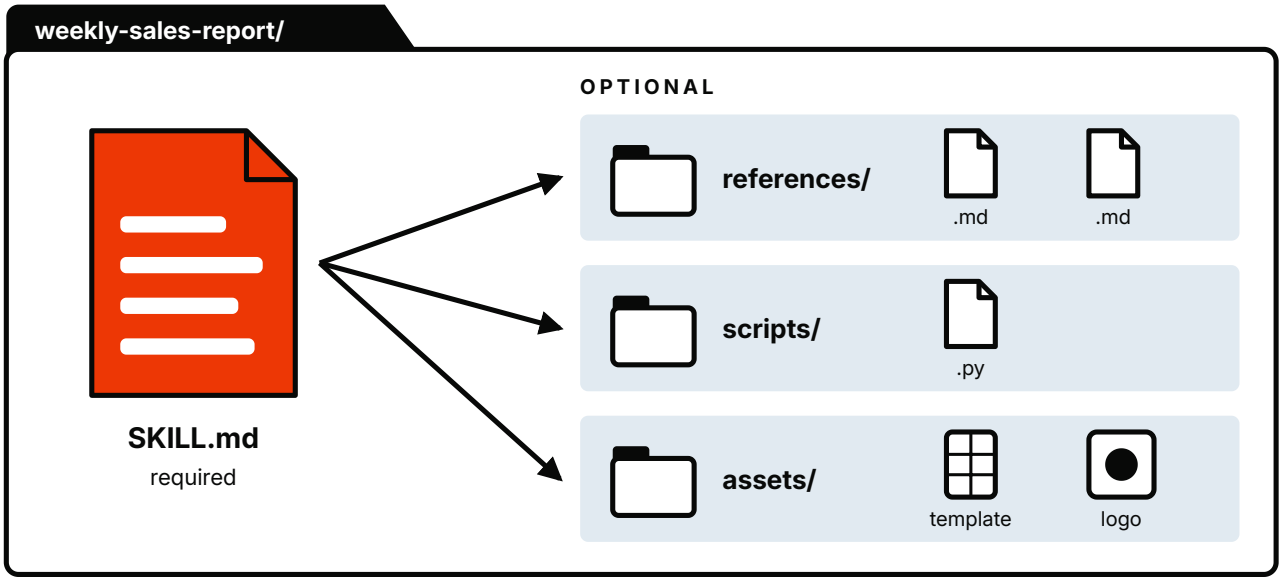
Checked against Anthropic's docs on 10 November 2026.

Alex Kerss

SECTION 1

What's inside a skill folder

A skill is a folder, not a file. The only file it has to contain is `SKILL.md`, and everything else is optional. Claude treats each part differently, so it's worth knowing what each one is for before you decide where a piece of your skill should go.



SKILL.md is the top-level file. It starts with a short header holding the skill's name and description, then the instructions underneath. This is the part Claude reads when the skill runs.

Reference files are extra markdown files, usually in a `references/` folder. They hold detail that Claude only needs some of the time, such as a full style guide or a list of product codes. `SKILL.md` points to them, and Claude opens one only when the instructions tell it to or when it decides it needs the detail.

Scripts are code files, usually in a `scripts/` folder. Claude runs them rather than reading them. Only what a script prints comes back into the conversation, not the code itself. That makes scripts a cheap way to do fixed steps, like checking a file or tidying a spreadsheet's columns, the same way every time.

Assets are files the skill uses in what it makes, usually in an `assets/` folder: templates, fonts, logos, sample data. Claude copies or fills them in. It doesn't need to read them to follow the instructions.

Scripts and some assets need **Code execution and file creation** switched on in **Settings > Capabilities** (shown as **Create and edit files** in the newer layout). Skills as a whole need it too. Without it, skills don't run at all in the Claude app.

Anthropic's guidance for laying out the folder is short. Keep the body of `SKILL.md` under 500 lines. Keep reference files one level deep, so `SKILL.md` points to each one directly rather than through another reference file. Give any reference file over 100 lines a contents list at the top.

SECTION 2

The header at the top of SKILL.md

The name and description aren't stored separately. They sit in a block of YAML at the very top of `SKILL.md`, between two lines of three dashes. The Claude app shows them in their own boxes, but they're part of the same file.



name

For the Claude app and the API, the name has strict rules:

- Up to 64 characters.
- Lowercase letters, numbers and hyphens only. No spaces and no capitals.
- No XML tags.
- It can't contain the words "anthropic" or "claude".

The Help Center's own example uses a friendly name with capitals and a space. Don't copy that. `brand-guidelines` passes every rule, while "Brand Guidelines" breaks two of them.

When you upload a zip, the folder inside it also needs the same name as the skill.

In Claude Code the name is optional. If you leave it out, the folder name is used, and that's also the command you type after the slash.

description

The description is required for the Claude app and the API. It has to be filled in, it can't contain XML tags, and it can be up to **1,024 characters**. It should say what the skill does and when Claude should use it, written in the third person.

The docs disagree on the limit. The Help Center says 200 characters, while the developer docs and the open standard say 1,024. The app enforces 1,024: a longer description won't save. A short one is safer for another reason, which I cover in the section on running lots of skills.

In Claude Code the description is optional, but without one Claude has nothing to match your message against. Claude Code also adds an optional `when_to_use` field, which it joins onto the end of the description.

The other fields

Skills you upload to your Claude account, send through the API or package with Anthropic's skill-creator can only hold six fields in the header:

FIELD	WHAT IT'S FOR
`name`	The skill's name, with the rules above
`description`	What the skill does and when to use it
`license`	The licence the skill is shared under
`compatibility`	What the skill needs to run, up to 500 characters
`metadata`	Your own notes as key and value pairs. Claude ignores them
`allowed-tools`	Tools the skill can use without asking you first

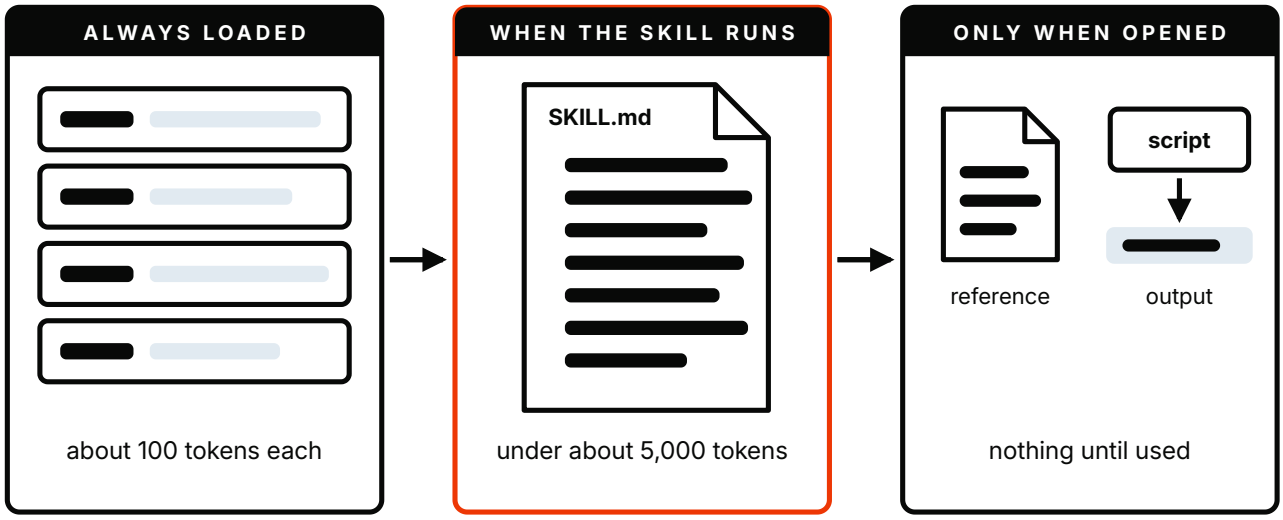
Claude Code accepts many more, such as `when_to_use`, `disable-model-invocation` and `hooks`. But if you add a Claude Code field to a skill and then upload it to your account, the upload fails with an error naming the field it didn't expect. Keep to the six above for any skill that needs to work outside Claude Code.

One more catch: if the header is badly formatted, Claude Code still loads the skill's body but treats the name and description as empty. You can still run it with the slash command, but Claude will never pick it by itself.

SECTION 3

How a skill loads into the context window

Claude doesn't read your whole skill until it needs it. A skill loads in three stages, and each stage costs a different amount of the context window.



Stage one: names and descriptions. At the start, Claude only sees the name and description of every skill you've switched on. Each one costs about 100 tokens. This is how Claude knows which skills exist and when each one fits. It's sent with every message, whether or not a skill runs.

Stage two: the body. When your message matches a description, or you pick the skill with **/** or **+**, Claude reads the body of `SKILL.md`. Anthropic's guidance is to keep the body under about 5,000 tokens.

Stage three: the rest of the folder. Reference files are read only when Claude opens them. Scripts are run, and only their output is added. Until then, they cost nothing.

That's why a skill can hold far more than its body. A long policy in a reference file costs nothing on the days Claude doesn't open it.

Two details are easy to miss. Once a skill's body has loaded, it stays in the conversation for the rest of it. In Claude Code, the file isn't read again on later turns, so an edit you make halfway through won't reach a skill that's already running. And because every enabled skill's description is sent with every message, switching on skills you never run still uses a little of your usage limits.

SECTION 4

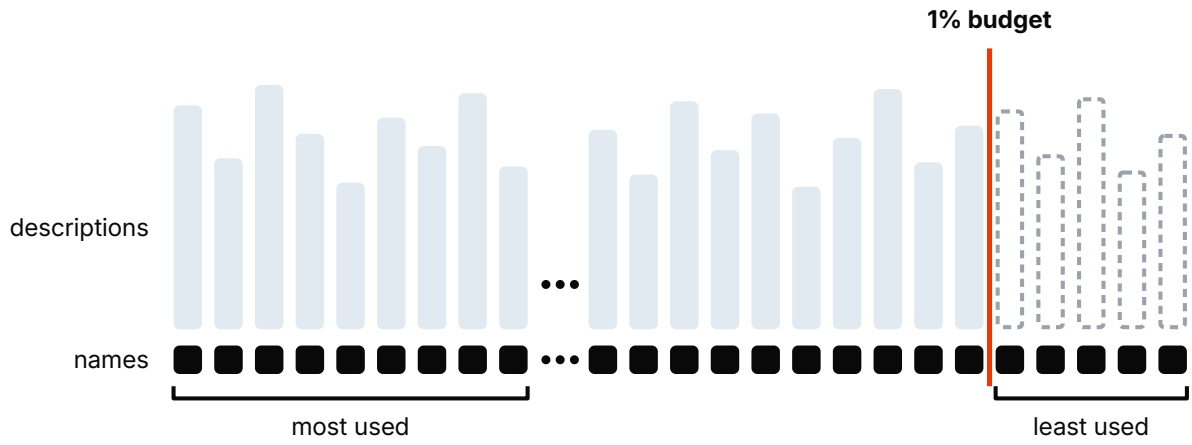
When you have lots of skills

Each description is small, but they add up. In Claude Code, this changes how skills behave once you have dozens of them, and long conversations add a second problem.

Descriptions get dropped in Claude Code

Claude Code puts every skill's name and description into one list, and that list gets a fixed share of the context window: **1%** by default. On a model with a 1 million token context window, that's roughly 40,000 characters for every skill's name and description together.

When the list doesn't fit, Claude Code drops descriptions, starting with the skills you run least. The name always stays, so you can still type `/weekly-sales-report`. But without its description, Claude has nothing to match your message against, so it may never pick that skill by itself.



There's also a cap on each entry. Claude Code cuts the description, plus any `when_to_use` text, at **1,536 characters**. Anything after that is never seen.

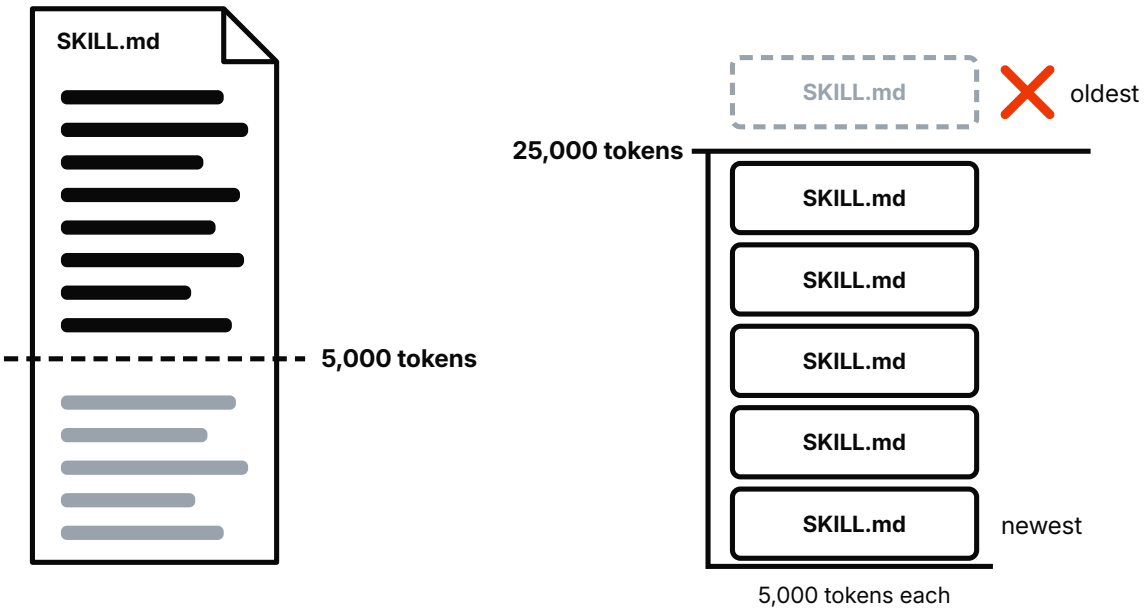
What helps:

- Keep descriptions short, and put the main use first.
- Switch off skills you don't need. Run `/skill-doctor` to see which skills you rarely run and what they cost, or `/doctor` for an estimate of the whole list.
- For a skill you only ever start yourself, add `disable-model-invocation: true` to its header. Claude won't pick it by itself, and its description stops taking up space in the list.
- To give the list more room, raise `skillListingBudgetFraction` in your settings (for example, `0.02` for 2%).

Long skills get cut short after compaction

When a Claude Code conversation gets too big, it's compacted into a shorter summary. Claude Code then brings back each skill you've run, but not all of it:

- Only the **first 5,000 tokens** of each skill come back.
- All skills together get **25,000 tokens** at most.
- It starts with the skill you ran most recently and works backwards, so if you've run lots of skills, the oldest can be left out entirely.



An instruction near the end of a long skill can quietly stop applying. Put your most important rules near the top of `SKILL.md`. As a rough guide, 12,000 characters is about 4,500 tokens, which keeps a skill inside the limit with some room to spare. If a skill seems to have stopped working after compaction, run it again to bring back the full text.

Reference files may not come back

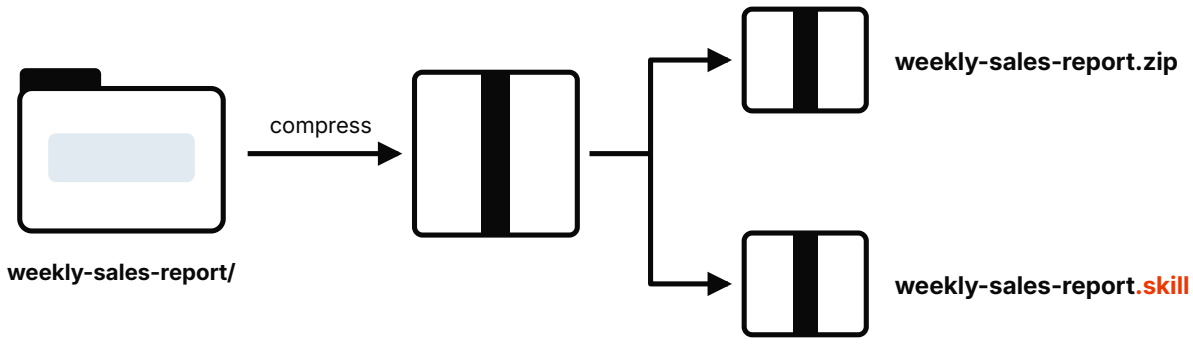
After compaction, Claude Code also re-reads a handful of recent files: up to five, picking the ones read or edited most recently. A reference file from your skill may not be one of them. If a rule has to hold every time, put it in `SKILL.md` near the top, and add a line telling Claude to re-read the reference file when it needs the detail.

The Claude app doesn't publish limits like these. Everything in this section is documented for Claude Code.

SECTION 5

What a .skill file is

A `.skill` file is a zip. It holds the whole skill folder, compressed, with a different file ending. Anthropic's skill-creator packages skills this way, and the Claude desktop app accepts a `.skill` file wherever it accepts a zip.



same contents, different ending

So the difference is only the name on the end. Both carry `SKILL.md` and every reference file, script and asset beside it. What sets a `.skill` file apart in practice is how Claude offers it to you, which I come to in the next section.

If you zip a skill yourself, zip the **folder**, not the files inside it. The zip should open to a single folder named after the skill, with `SKILL.md` inside that folder. A zip of loose files won't upload. A single `.md` file on its own isn't a documented way to upload a skill either.

To upload a zip or `.skill` file in the Claude app:

1. Click **Customize** in the sidebar and open the **Skills** tab.
2. Click **Add** and choose **Upload skill**.
3. Pick your `.zip` or `.skill` file.

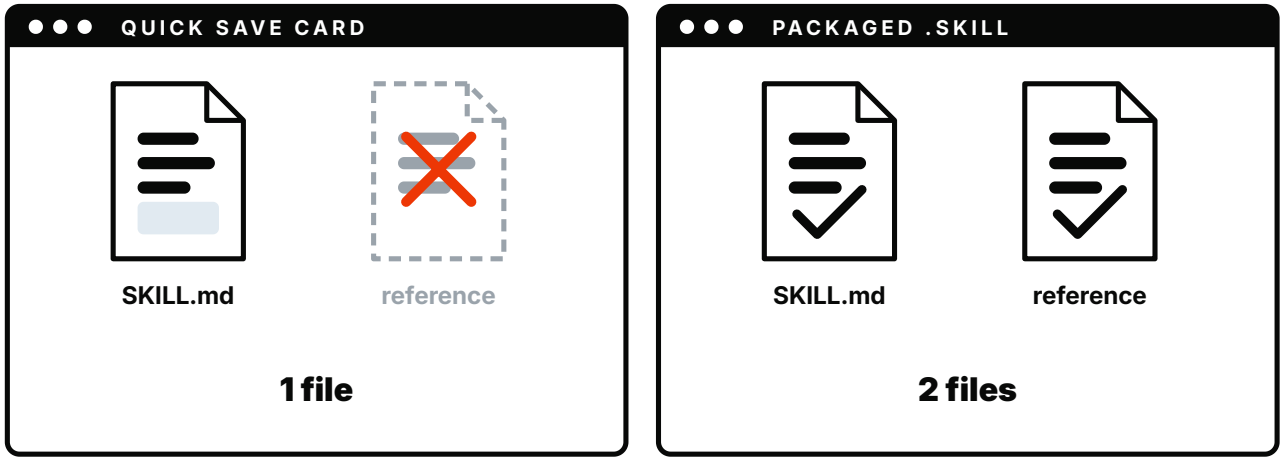
••• SCREENSHOT 1

Customize > Skills in the Claude desktop app with the Add menu open, showing Upload skill.

SECTION 6

The quick save card vs a packaged skill

When you ask Claude in the Claude app to make a skill, it usually offers a **quick save card**: a small card in the conversation with a **Save** button. It's quick, but the card only saves `SKILL.md`.



In my tests in the Claude desktop app on a Mac in November 2026, this caused two problems:

- **Making a skill with reference files.** Claude wrote the skill, wrote the pointer to the reference file and even wrote the reference file itself. But the save card couldn't save the reference file, so Claude folded its contents into `SKILL.md` instead.
- **Changing a reference file later.** Asking Claude to edit a reference file in a skill you've already saved fails the same way, because the save card can only change `SKILL.md`.

A packaged skill doesn't have either problem. Claude builds the folder as files, packages it as a `.skill` file and gives you a file card. Clicking it saves or updates the whole skill, so every file comes through. When it updates a skill you already have, the change shows as a new version.

The quick save card is fine for a skill that's only ever one file. For anything with reference files, ask for a packaged skill in so many words. Something like this works:

> Don't use the quick save card. Build the skill as files, a SKILL.md and a references folder, and package it as a .skill file.

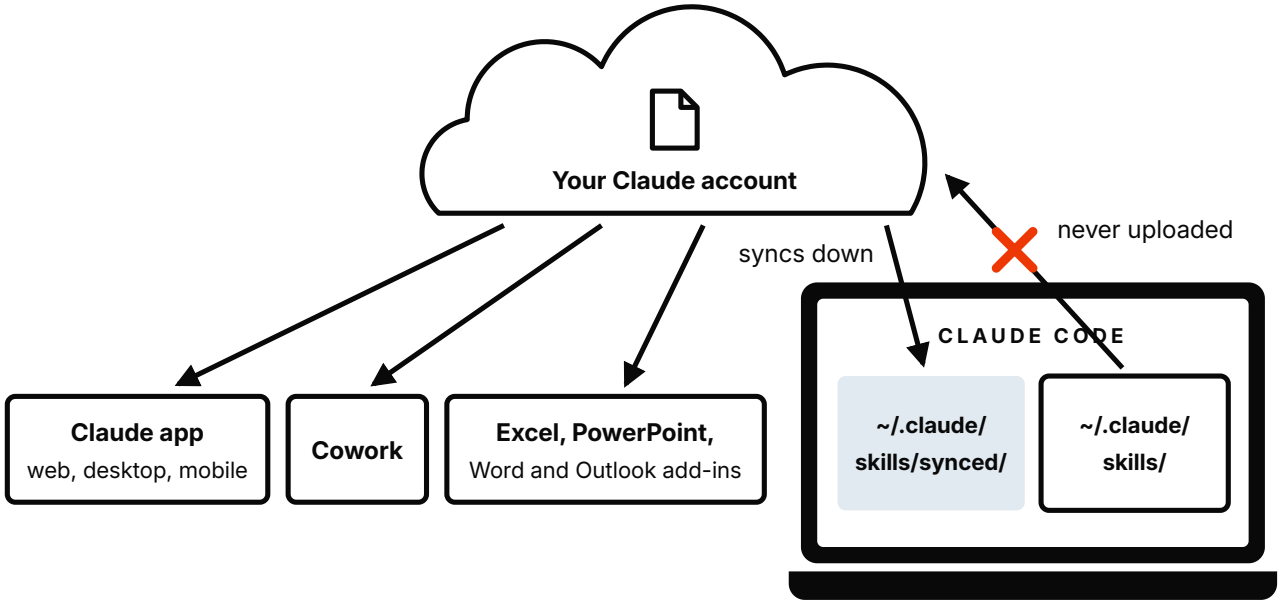
To change a reference file later, ask the same way: tell Claude not to use the quick save card, to build the whole skill again and to package it as `your-skill-name.skill`.

You can also edit a skill's files yourself, next to the conversation. Highlight some text and click **Edit with Claude**.

SECTION 7

Where your skills are stored

Skills are kept in two separate places, depending on where you made them.



Your Claude account. A skill you save in the Claude app is stored in your account, in the cloud, not on your computer. That's why the same skill turns up on the web, in the desktop app and on mobile, and in Cowork. Skills you've switched on also work in the Claude add-ins for Excel, PowerPoint, Word and Outlook (type **/**).

Folders on your computer. Claude Code reads skills from folders on disk:

KIND	WHERE IT'S KEPT	WHO GETS IT
Personal	`~/.claude/skills/<skill-name>/SKILL.md`	You, in every project on that computer
Project	`.claude/skills/<skill-name>/SKILL.md` inside a project	Anyone who opens that project
Synced from your account	`~/.claude/skills/synced/`	You, when signed in with your Claude account

Claude Code watches these folders, so adding, editing or removing a skill takes effect in the running session without a restart.

The sync only goes one way

When you sign in to Claude Code with your Claude account, it downloads the skills switched on in that account into the `synced` folder. It does this in the background when a session starts. After that, it checks for changes about every 10 minutes while you're working and about every 40 minutes while the session sits idle. Skills you add, change or switch off in the app are added, updated or removed without a restart.

This needs **Claude Code v2.1.273 or later**. It doesn't happen if you sign in with an API key, run Claude Code through Amazon Bedrock, or start in bare mode or safe mode.

Nothing goes back up. That has three effects:

- If you edit a synced skill in Claude Code, the edit isn't saved to your account, and the next sync can overwrite it. Change the skill in the Claude app instead.
- The Claude app keeps running the version in your account, not your edited copy.
- A new skill made in Claude Code never reaches your account, so the Claude app can't see it. To move it across, zip the skill folder and upload it under **Customize > Skills**.

If a synced skill has the same name as one of your own Claude Code skills, `/name` runs yours, and the synced one runs as `/anthropic-skills:name`.

To stop the sync on one computer, set `syncClaudeAiSkills` to `false` in your Claude Code user settings.

Plans

The Help Center lists skills on **Free, Pro, Max, Ultra, Team and Enterprise**, with code execution switched on. The developer docs leave Free out when it comes to uploading your own skills, so if you're on Free and the upload option is missing, that's why.

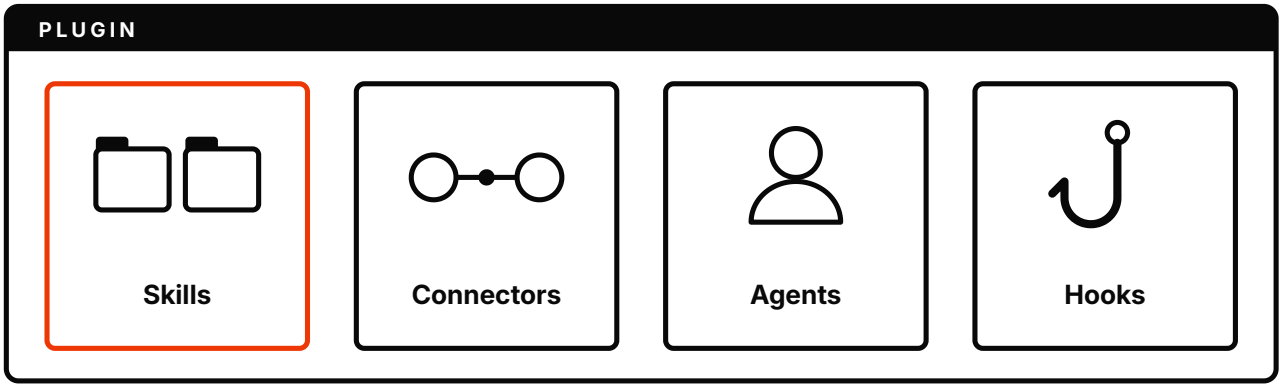
On **Team**, skills are on by default. On **Enterprise**, an owner has to switch on both code execution and Skills first. Sharing a skill with colleagues inside the app is only on Team and Enterprise. On the other plans, each person uploads skills to their own account.

The API keeps its own separate store of skills, shared across your whole workspace. It allows up to 20 skills in one call to the API, and up to 30 MB per skill.

SECTION 8

How skills relate to plugins

A skill is one set of instructions. A **plugin** is the packaging around one or more of them. It can bundle skills together with connectors, agents and hooks, so a whole set of tools installs in one go.



In the Claude app, skills, connectors and plugins each have a tab on the **Customize** page. Anthropic is replacing those tabs with one **Plugins** page, rolling out gradually from 29 October 2026, so some apps already show it and others still show Customize. Your existing skills stay installed either way. Anthropic has said skills will be packaged into plugins over time. Public plugins carry a **Verified** or **Community** label.

A few things carry over from what you know about skills:

- **Plugin skills cost context too.** Every enabled plugin's skill descriptions are loaded on every turn and count toward your usage, whether or not you run them.
- **Plugin skills are kept apart by name in Claude Code.** They run as `/plugin-name:skill-name`, so they never clash with your own skills.
- **Removing a plugin doesn't disconnect its connector.** Disconnect the connector separately.
- **Admins should know the link.** From 30 November, if Skills are switched off for a Team or Enterprise organisation, plugins are also blocked in the Claude desktop app.

If you have a skill both on its own and inside a plugin, switch one of them off. Two skills with overlapping descriptions compete for the same messages.

SECTION 9

Data handling and safety

Skills don't get the same data protections as some other parts of Claude.

- **Skills aren't covered by zero data retention.** If your organisation has a zero data retention agreement, it doesn't extend to skills.
- **Skills aren't HIPAA-eligible.** On a HIPAA setup, the business associate agreement doesn't cover skills, plugins or connectors. Keep health data out of them.

A skill can also do more than an ordinary prompt can. It can run code, and it can install third-party packages or tell Claude to. Anthropic's advice is to use skills only from trusted sources, meaning ones you wrote yourself or got from Anthropic, and to treat installing a skill like installing software. Before you install one from anywhere else, read every file in it, including the scripts.

A few more points to bear in mind:

- **Never put passwords or API keys in a skill.** Anyone you share the file with gets them too.
- **Deleting a skill is permanent.** The only way back is to upload the file again, so download a copy first.
- **Admins can see some of it.** On Team and Enterprise, admins can see the names and sharing status of your own skills, but not what's in them.
- **Enterprise can scan skills.** With security scanning switched on (in beta since August 2026), a skill from someone else can't run until it passes a scan. Most scans take one to two minutes.
- **Shared skills arrive switched off.** On Team and Enterprise, a skill someone shares with you is view-only and stays off until you turn it on. When the owner changes it, your copy updates automatically.

AT A GLANCE

Quick reference

••• QUICK REFERENCE

Folder: `SKILL.md` (required) plus optional `references/`, `scripts/` and `assets/`. Keep the body under 500 lines.

name: up to 64 characters. Lowercase letters, numbers and hyphens. No XML tags, and no "anthropic" or "claude". The zipped folder needs the same name.

description: required in the app and API. Up to 1,024 characters (the app enforces this, though the Help Center says 200). Say what it does and when to use it.

Header fields allowed in uploads: `name`, `description`, `license`, `compatibility`, `metadata`, `allowed-tools`. Anything else fails.

Loading: names and descriptions always (about 100 tokens each), the body when the skill runs (under about 5,000 tokens), other files only when opened.

Claude Code limits: skill list gets 1% of the context window, least-run descriptions dropped first. Each description cut at 1,536 characters. After compaction, the first 5,000 tokens per skill, 25,000 in total, newest first.

.skill file: a zip of the skill folder with a different ending.

Quick save card: saves `SKILL.md` only. Ask for a packaged `.skill` file whenever reference files are involved.

Storage: the Claude app saves to your account. Claude Code reads `~/.claude/skills/` and `.claude/skills/`.

Sync: account to Claude Code only, v2.1.273 or later, every 10 minutes or so while working. Never back up.

Plugins: bundles of skills, connectors, agents and hooks, under Customize > Plugins (one Plugins page as the rollout reaches you).

Data: not covered by zero data retention, not HIPAA-eligible. Only install skills from sources you trust.