CLAUDE PROJECTS · GUIDE 3 OF 3

Claude Projects Advanced Patterns

How to set up projects so they still work well after months of use.

Checked against Anthropic's docs on 10 November 2026.

Alex Kerss

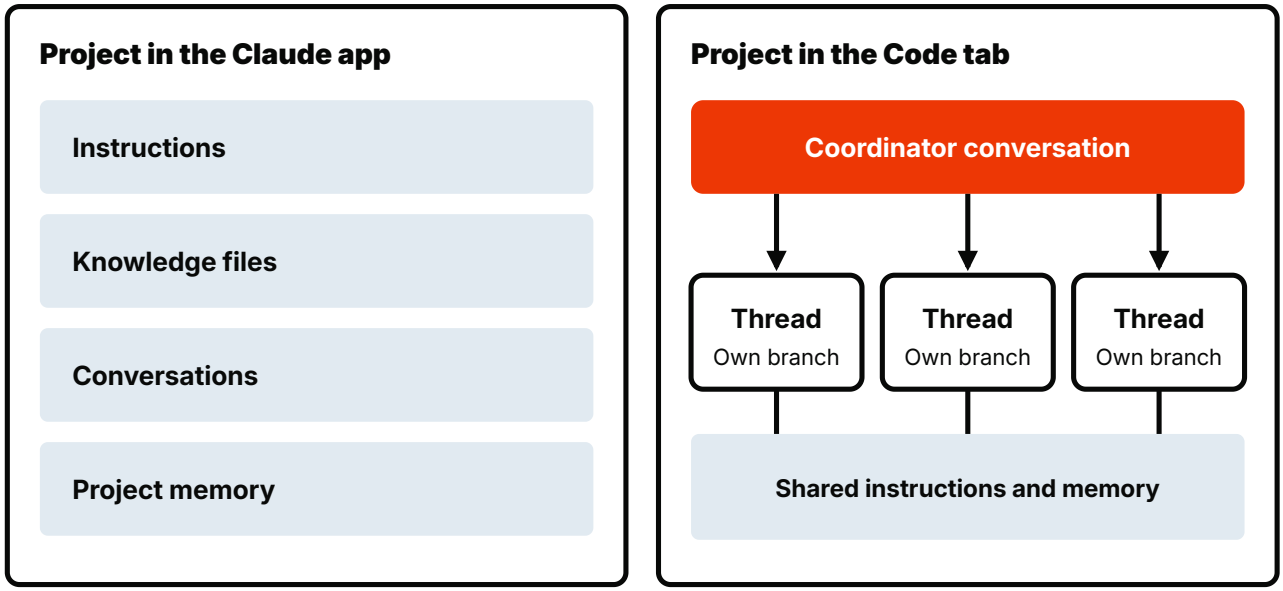
START HERE

Two kinds of project

There are now two different things called a project, and the patterns depend on which one you're in.

The first is the project in the Claude app, on the web, desktop and mobile. It holds instructions, knowledge files, its own conversations and its own memory. Since Cowork became part of one Claude, Cowork projects are the same kind. Projects are on every plan, though Free accounts can have at most five.

The second is the redesigned Project in the Code tab of the desktop app and at claude.ai/code. It's one coordinator conversation that starts threads, and each thread is a separate session that usually runs in the cloud. It's a public beta on Pro, Max and Ultra, rolling out gradually, and it isn't offered on Team or Enterprise yet. The projects in the Claude app stay on the older design for now.



The first six patterns are about projects in the Claude app. The last three are about the Code tab. The clicks are in the hands-on guide, and the limits and memory mechanics are in the under-the-hood guide.

PATTERN 1

One project per ongoing area of work

Conversations in a project share only what's set at project level:

- the instructions
- the knowledge files
- the project memory

Everything else stays inside the conversation where it happened. So if one conversation works out the new price list, a second conversation in the same project won't know about it unless you add it to the project's knowledge.

That's why the best way to split your work is by area, not by piece of work. An area is something you keep coming back to over weeks, with the same background each time. Your customer support inbox is an area. A single refund email is not.

When to use it. Make a project when you notice you're pasting the same background into new conversations. Keep one-off questions outside projects, so they don't add to any project's memory.

Example. You run two holiday lets. One project holds the house rules and your standard replies, and you start a new conversation in it each time a guest writes in. When a reply turns out well, add it to the project's knowledge so the next conversation can find it.

One more habit: put anything Claude needs to know in the instructions or files, never in the project's name or description. Claude can't see either of those.

PATTERN 2

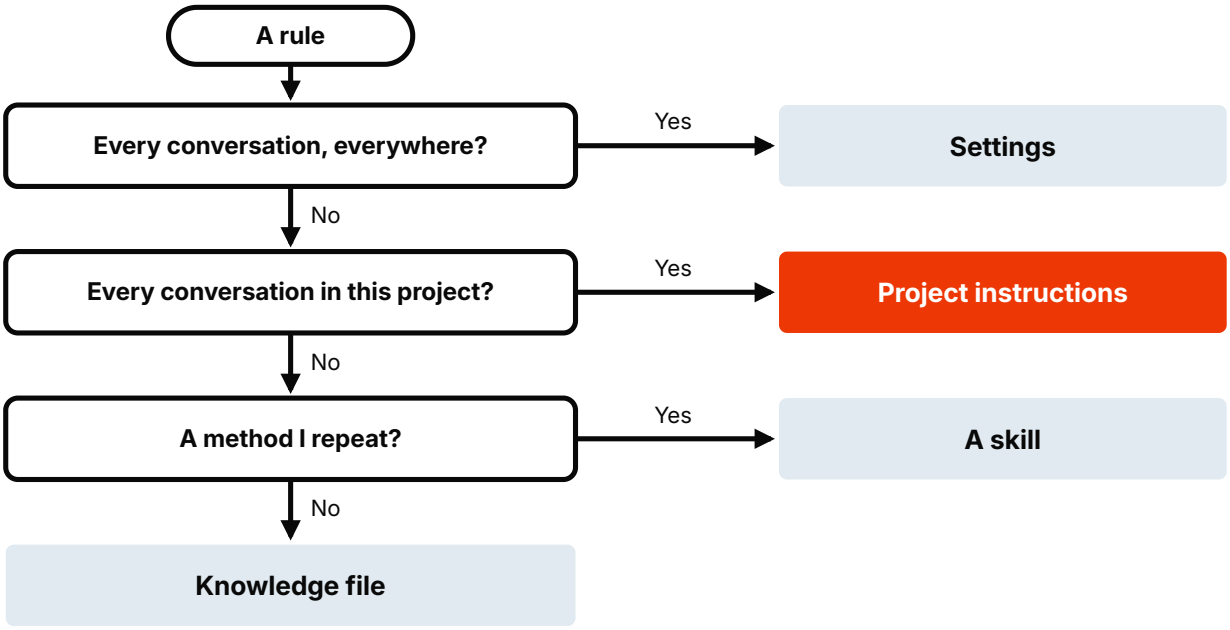
Instructions that hold up

Project instructions apply to every conversation in the project. That makes them the right place for anything that should be true every time, and the wrong place for anything that's only true some of the time.

The test I'd apply is how often Claude needs it.

PUT IT IN	WHEN IT'S	EXAMPLE
Project instructions	Needed in every conversation in this project	"You're helping me reply to guests. Write in British English and keep replies under 150 words."
A knowledge file	Reference material Claude looks things up in	The house rules, the price list
A skill	A method you repeat, here and in other projects	How to turn a call recording into minutes with actions
Your custom instructions in Settings	True for every conversation you have, in any project	Your name, your time zone, how you like answers laid out

Anthropic describes a project as background that's always loaded, and a skill as a procedure that loads only when the work calls for it. So if a method keeps turning up in several projects, move it into a skill and keep each project's instructions short.



What good instructions cover. Anthropic's guidance for Code-tab Projects gives a useful shape, and it works just as well in the Claude app:

1. What the project is for.
2. Where the material comes from and what to trust first.
3. How Claude should check its work before it hands it over.
4. How to handle something missing.
5. What needs your go-ahead first.

Point 4 stops guessing. A line such as "If a price isn't in the price list, say so and stop" gets you a short question back instead of a made-up number.

When to use it. Rewrite the instructions whenever you correct Claude for the same thing twice. If the correction only applies to one kind of document, it probably belongs in a file or a skill instead.

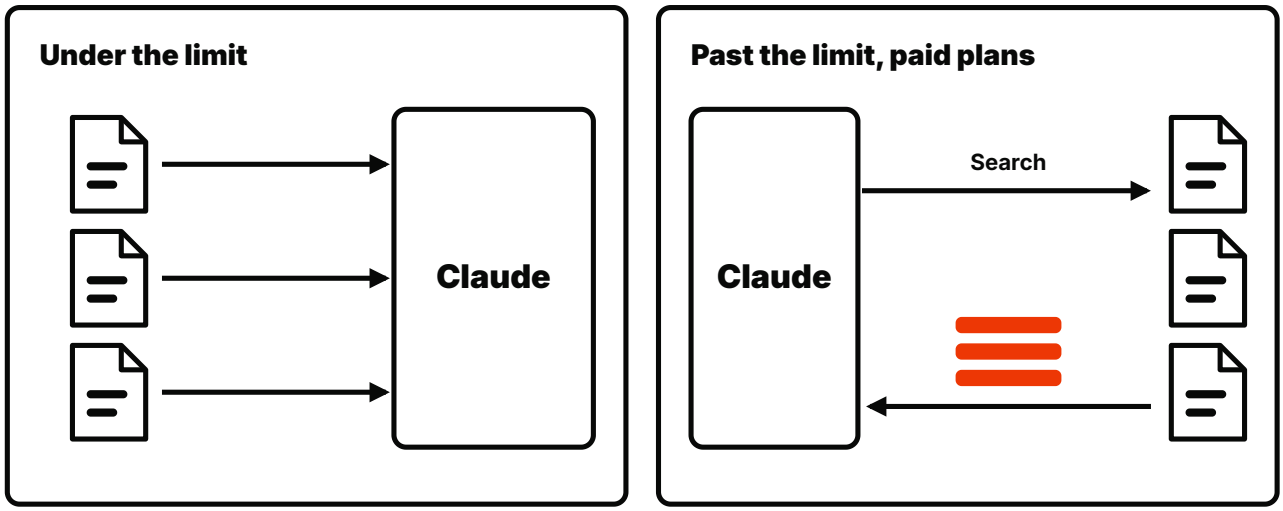
Example. For a weekly sales summary, the instructions say who reads it and how long it should be. The spreadsheet column definitions sit in a knowledge file, because Claude only needs them when it's reading the numbers.

PATTERN 3

A small, current knowledge base

By default, Claude reads all of a project's knowledge with every message. That's why a few current files work better than a big pile of old ones. If last year's price list and this year's are both in the project, Claude sees both, and you're relying on it to pick the right one.

On paid plans, Pro, Max, Ultra, Team and Enterprise, this changes once the knowledge gets close to the context window limit. The project switches on its own to retrieval, which Anthropic calls RAG. Claude then searches the knowledge with a project knowledge search tool and pulls in only the passages that look relevant to your question. That can expand a project's capacity by up to ten times. You can't switch it on or off yourself, and if the knowledge shrinks again, the project can go back to reading everything. On the Free plan there's no retrieval, so the knowledge has to fit in the context window.



Once a project is past that point, how you name and group files starts to show in the answers. Anthropic's tips:

1. Give files clear, descriptive names, such as "Price list 2026" rather than "final v3".
2. Keep related documents together in the same project.
3. Name the document you mean in your question.

Replacing stale files. When a document changes, delete the old file and upload the new one, rather than adding the new version alongside. Project files can be up to 30 MB each. Claude reads the text in them, and for PDFs of 100 pages or fewer it looks at the pictures and charts too.

If a document changes every week, Google Drive can save the re-uploading, because Drive files in a project stay in sync. That only works in private projects.

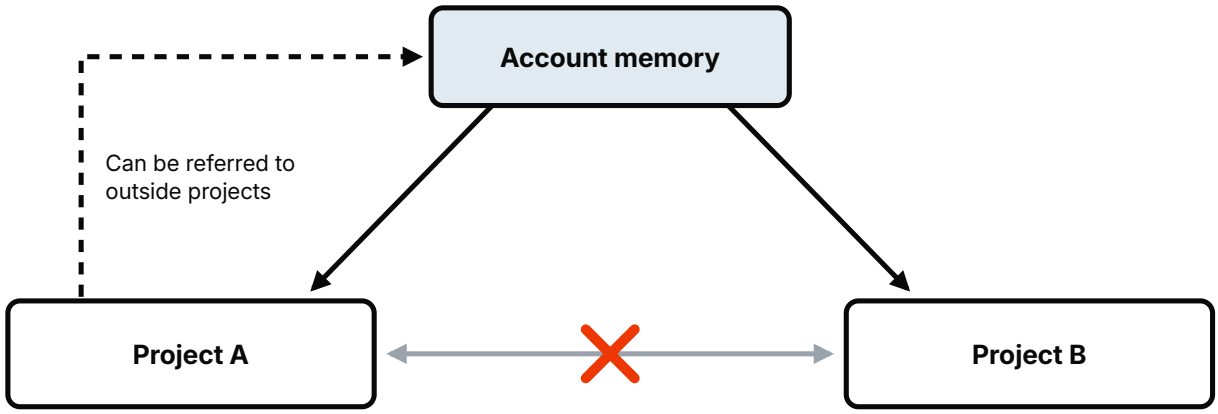
When to use it. Do a tidy-up whenever answers start quoting something out of date. Ask yourself which files Claude would need if you were starting the project today, and remove the rest.

Example. A project for answering staff questions about HR policy. Each policy is one file named after the policy. When the holiday policy is updated, the old file comes out on the same day the new one goes in.

PATTERN 4

Clean memory, and when to isolate a project

Each project has its own memory, but by default it isn't sealed off. Inside a project, Claude draws on that project's memory and on your account memory from conversations outside projects. It doesn't draw on the memory of other projects. Anything new that it remembers is saved to the project, and if memory is on, Claude can refer to it in conversations outside projects too.



That default suits most areas of work. It suits a sensitive project less well, such as a client under a confidentiality agreement.

To isolate a project, turn off **Use account memory** under **Memory** on the project's page. Your account memory then stays out of it, and the project keeps its own memory.

••• SCREENSHOT 1

a project's page in the Claude app, with the Memory section open and the Use account memory switch visible.

To keep one conversation out of memory, turn off **Memory** in the **+** menu before you send the first message. That conversation then neither reads from nor adds to the project's memory.

On Free, Pro, Max and Ultra, memory is on by default. On Team and Enterprise, the owners decide.

When to use it. Switch off account memory for any project where you'd be uncomfortable seeing its details turn up somewhere else. Switch off memory for single conversations where you're testing ideas you don't want remembered.

Example. You keep a project for a client's contract negotiation. With account memory off, nothing from your other work comes in. A quick "what if we offered 60-day terms" conversation runs with memory switched off, so the idea isn't saved as if it were agreed.

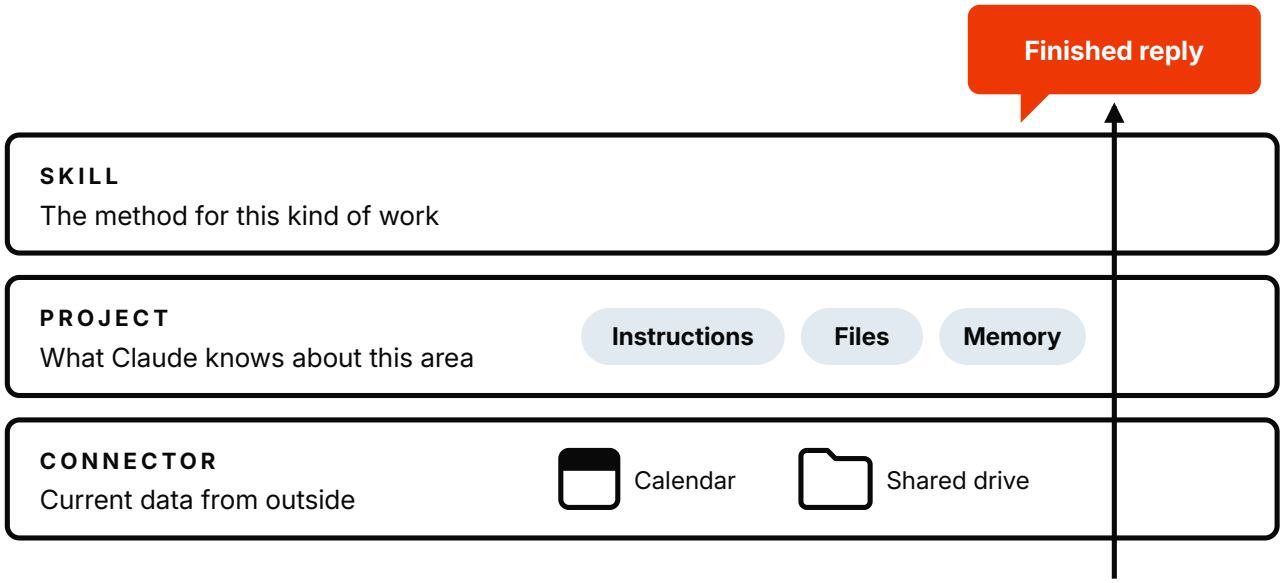
PATTERN 5

Projects, skills and connectors together

These three do different things, and they work best combined.

- **A project** holds what Claude should know about one area of your work.
- **A skill** holds the method for one kind of work, and loads only when the work calls for it.
- **A connector** links Claude to an outside service, so it can read or act on current data there.

Anthropic puts the last two together like this: connectors give Claude access to tools, while skills teach Claude how to use those tools well. Skills and connectors both live on the **Customize** page, each with its own tab. The skills and connectors turned on for your account are available inside your projects as well.



When to use it. Reach for a skill when the same method shows up in more than one project. Reach for a connector when the files in your project keep going out of date because the source changes daily.

Example. A skill that writes a monthly sales report in your standard layout works in any project. Inside the project for your shop, it picks up the product list and the notes on last quarter. With your spreadsheet connected, it can read this month's figures instead of you uploading a copy.

PATTERN 6

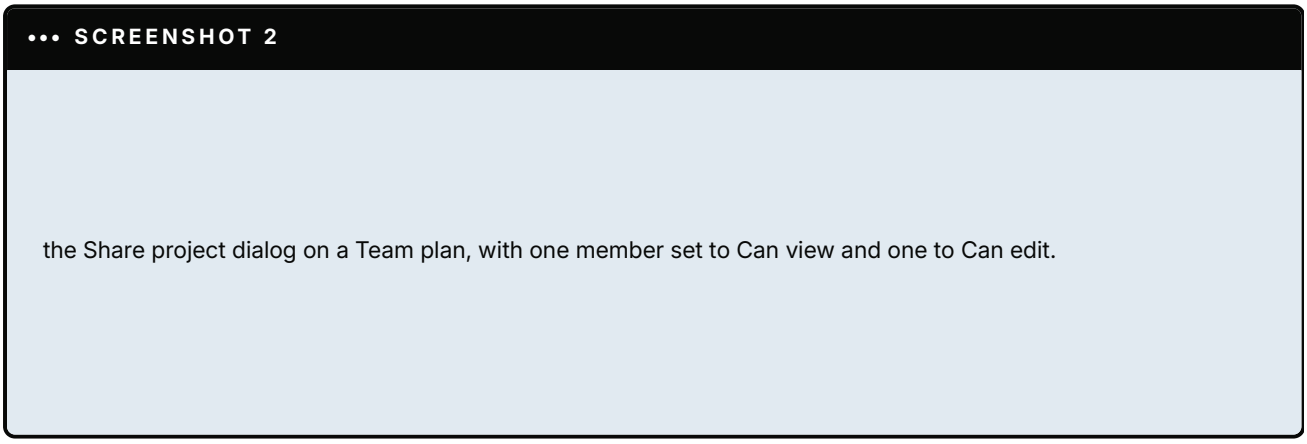
Team projects on Team and Enterprise

Sharing a project is only on Team and Enterprise. When you create a project there, you choose whether it's private or shared with the organisation. You can also share later through **Share project**, adding people by name or email, or the whole organisation.

There are two permission levels.

- **Can view**: people see the contents, knowledge and instructions, and can start their own conversations in the project, but can't change it.
- **Can edit**: people can also change the instructions and knowledge and manage members. On Enterprise they can add project tags, which are in beta.

••• SCREENSHOT 2



the Share project dialog on a Team plan, with one member set to Can view and one to Can edit.

A shared project works best when one or two people look after it. Give **Can view** to most members and **Can edit** only to the people who maintain it. Otherwise one edit to the instructions changes every colleague's answers without anyone noticing.

A few more points for shared projects:

1. Owners can turn off project sharing for the organisation. Existing shares stay in place.
2. Sharing settings can't be changed on mobile.
3. Google Drive can't be added to a shared project, so shared knowledge has to be uploaded files.

When to use it. Share a project when several people answer the same kind of question from the same documents.

Example. A sales team shares one project with the product sheets and pricing rules. The sales manager has Can edit and updates the pricing file each quarter. Everyone else has Can view.

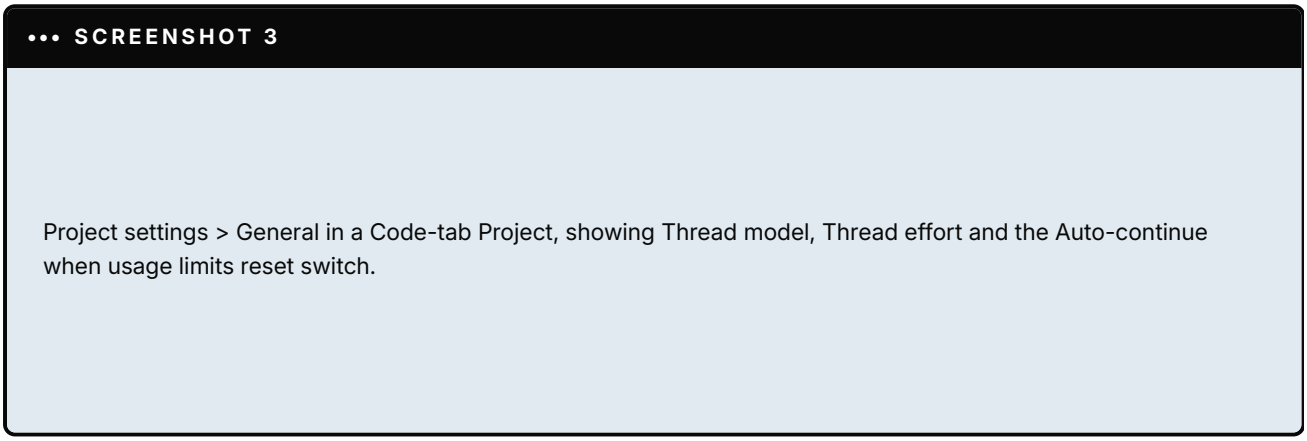
PATTERN 7

Code-tab Projects, keeping usage under control

A Code-tab Project uses your plan faster than a single session does. On Pro especially, Anthropic says to expect to reach your limit sooner on days you run one. Most of that comes from defaults you can change.

Lower the defaults first. A new project runs Opus everywhere, at high effort for threads and low effort for the coordinator conversation. Change **Thread model** and **Thread effort** in **Project settings > General** before you send work, unless the work needs the largest model. To use a different model for one piece of work, ask for it when you send it.

••• SCREENSHOT 3



Project settings > General in a Code-tab Project, showing Thread model, Thread effort and the Auto-continue when usage limits reset switch.

Watch auto-continue. A thread that stops at your usage limit waits, then carries on by itself when the limit resets, so its work counts against your next window. Turn off **Auto-continue when usage limits reset** in **Project settings > General** if you'd rather decide. It applies to the whole project.

Watch for wake-ups. A thread that opened a pull request watches it. When CI fails or a review comment arrives, an idle thread wakes up and uses your plan again. Ask the thread to stop watching the pull request when you no longer want that.

Know how to stop things.

1. **Stop**, or Esc, interrupts one thread while it's working.
2. **Pause**, in Project settings > General, stops every thread and the coordinator at once, and scheduled work doesn't run until you resume.
3. **Archive** hides the project and archives its threads, which stops any that were running or watching a pull request.

A project with no running threads, no watched pull requests and no new messages uses nothing while it sits idle.

Start fresh rather than reviving. On Pro, Max and Ultra, the cache lasts an hour. A thread idle for longer re-reads its whole conversation before it does anything. For new work, ask for a fresh thread instead of sending more to a big old one.

Two more things. On your first project, Claude takes a turn of its own as soon as the project is created, unless you send a message first, and that turn uses your plan. And "run at most two threads" is a preference Claude follows, not a cap. **Usage** in Project settings shows where the tokens went, by thread and by model.

Example. You're clearing a backlog of small bugs on a web shop's checkout. Threads run on a smaller model at medium effort, auto-continue is off, and you ask the coordinator to answer quick questions itself rather than starting a thread for each one.

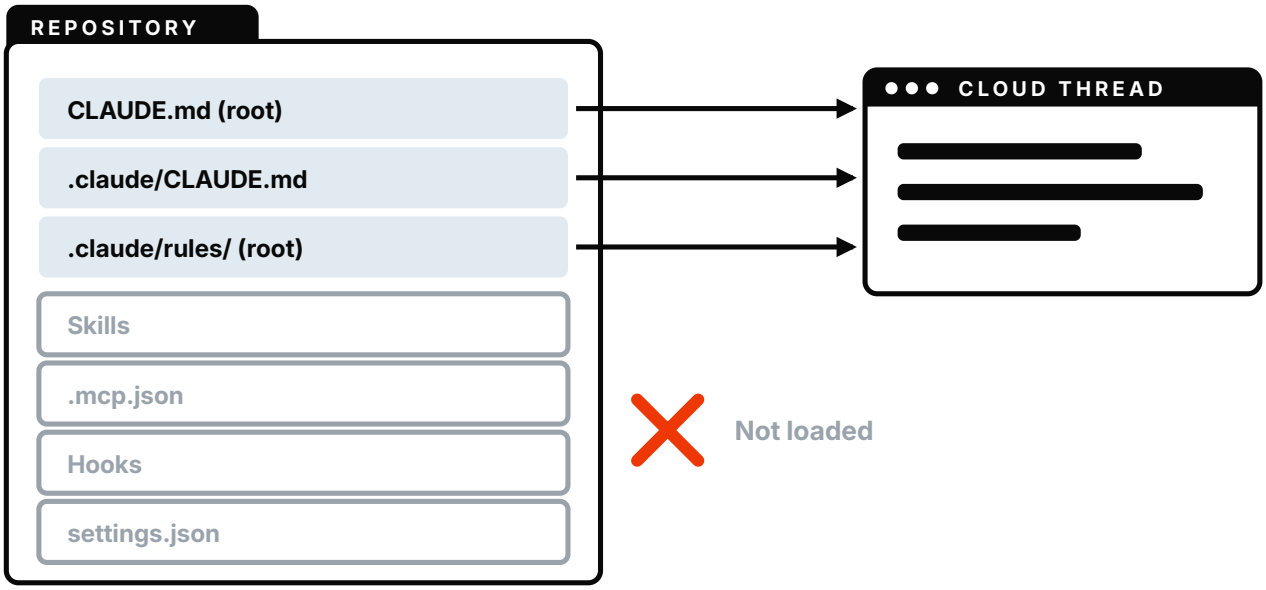
PATTERN 8

Put the rules threads need in CLAUDE.md

Cloud threads don't bring your repository's full Claude Code setup with them. From each repository in the project, a thread loads only these at the start:

- `CLAUDE.md` at the repository root
- `.claude/CLAUDE.md`
- `.claude/rules/` at the repository root

Nothing else in the repository's `.claude/` setup loads. That includes its skills and its `.mcp.json` servers, and everything in `.claude/settings.json`, such as hooks and permission rules. `CLAUDE.md` files in subfolders aren't loaded either.



So any rule a thread must follow has to be written where the thread will read it.

Split the rules by scope.

- **Project instructions**, in **Project settings > Memory > Project instructions**, up to 16,000 characters, are the brief every new thread starts from. Put the rules here that apply across repositories, such as what needs your go-ahead.
- **A repository's** `CLAUDE.md` holds the rules for that repository, such as its build and test commands. Anthropic recommends keeping each one under 200 lines.
- **Project memory** holds what Claude learns along the way. When you correct a thread, tell Claude to remember the correction, and later threads start with it.

Replace what doesn't load. Hooks and permission rules don't run in cloud threads, so a rule that a hook used to enforce locally has to be written as an instruction. Bear in mind that `CLAUDE.md` is context Claude follows, not a setting it enforces. For the rest:

1. **Skills**: turn them on for your claude.ai account, and cloud threads load them.
2. **Plugins**: add them in **Project settings > Plugins**.
3. **MCP servers**: connect them as connectors on your claude.ai account. The coordinator conversation has no connectors, so anything that needs one goes to a thread.

Changes to instructions, plugins and repositories reach new threads, not the ones already running.

When to use it. Check what's in your repository's `.claude/` folder before you add it to a project. Anything there that threads need, apart from `CLAUDE.md` and the root rules, has to come in another way.

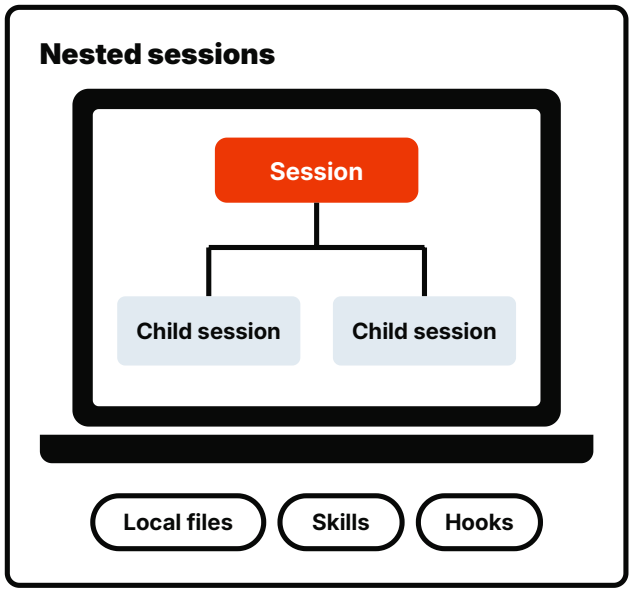
Example. Locally, a hook stops Claude editing the payments config. In the project, that becomes a line in `CLAUDE.md`: "Never edit config/payments.yml. If a change needs it, stop and say so in the thread."

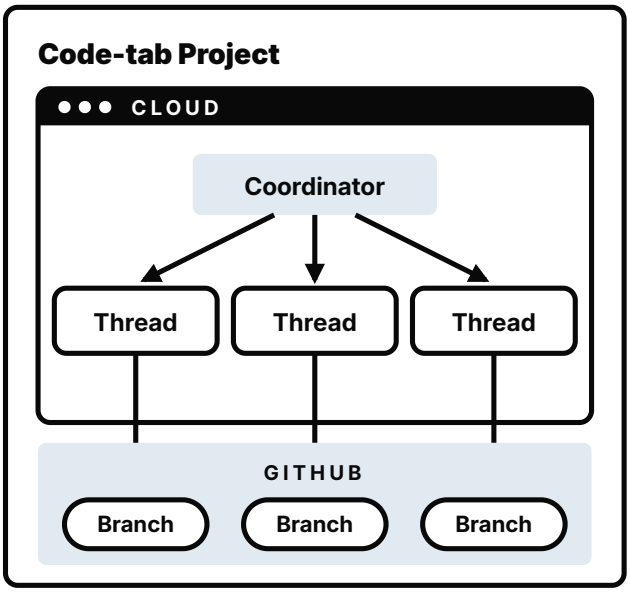
PATTERN 9

Nested sessions instead of a project

Nested sessions are a separate Code-tab feature, added on 15 October 2026. A session starts another session for a separate piece of work, and the new one appears under the original. Each has its own context, so the first session doesn't fill up. In **Settings > Claude Code** you choose what happens when you ask for something unrelated in the middle of a session.

The difference that decides it is where they run. Nested sessions work in local and SSH sessions, on your own computer or a server you connect to. They get your whole local setup, hooks and MCP servers included. A Code-tab Project runs its threads in the cloud with the limited setup in Pattern 8.





When nested sessions fit better.

1. Every piece of the work needs your computer, such as a local database or a device emulator.
2. You want to watch and steer each piece as it runs, then finish.
3. You're on Team or Enterprise, where Code-tab Projects aren't offered yet.
4. Your code isn't on github.com. Projects need it there, with the Claude GitHub App installed.

A project fits better when the work has a goal that outlasts one session and keeps producing new pieces, or when you want it to carry on with your laptop closed.

Nested sessions have no shared memory of their own, so for work that goes on for weeks, keep a folder with a good `CLAUDE.md`. Anthropic doesn't say which plans get nested sessions, but the Code tab itself needs a paid plan.

Example. You're updating an app that talks to a database on your laptop. One session plans the change, then starts one nested session for the database migration and one for the screens that use it. You check each, and the whole thing is done that afternoon.

GOOD TO KNOW

When not to use a project at all

A project adds standing context and, in the Code tab, ongoing cost. Some work is better without one.

- **A one-off question.** Ask it outside any project, so it doesn't pick up project memory or add to it.
- **Something you want true in every conversation.** Put it in your custom instructions in Settings.
- **A method you repeat across different areas.** Make it a skill.
- **One piece of code work that fits in a session.** Start a cloud session or a local one yourself.

- **Something that repeats on a schedule with no conversation around it,** such as a report every Monday. Anthropic suggests a scheduled run on its own rather than a project.
- **Work where every piece needs your computer.** Use local sessions or nested sessions.

AT A GLANCE

Quick reference

••• QUICK REFERENCE

Projects in the Claude app

- Claude can't see the project's name or description. Put everything it needs in the instructions or files.
- Conversations in a project share only the instructions, the knowledge and the project memory.
- Paid plans switch to retrieval automatically near the context limit, up to 10x capacity. Name files clearly and name the file in your question.
- Replace stale files; don't add new versions alongside old ones.
- Isolate a project: **Use account memory** off, under **Memory** on the project page.
- Keep one conversation out of memory: **Memory** off in the **+** menu, before the first message.
- Sharing: Team and Enterprise only. **Can view** for most, **Can edit** for the maintainers.

Code-tab Projects (beta: Pro, Max, Ultra)

- Defaults: Opus everywhere, high effort for threads. Change in **Project settings > General**.
- **Auto-continue when usage limits reset** is on by default. Same place to turn it off.
- Idle threads wake on CI failures and review comments. Ask the thread to stop watching the pull request.
- **Stop** for one thread, **Pause** for everything, **Archive** to stop and hide.
- Threads load only root `CLAUDE.md`, `.claude/CLAUDE.md` and root `.claude/rules/`. Skills from your account, plugins from Project settings, MCP through account connectors.
- Project instructions up to 16,000 characters. Repository rules in `CLAUDE.md`, under 200 lines.
- Every piece of the work needs your computer? Nested sessions, in local and SSH sessions.