Build Your Own AI Skill

Creating and improving skills

Adrien Matray
(Federal Reserve Bank of Atlanta)

Inter-American Development Bank, November 2026

Any opinions expressed are those of the author(s) and do not represent the views of the Federal Reserve Bank of Atlanta or the Federal Reserve System.

Before we start: unzip the package and open your project

1. Download `skills_pack.zip`:
   `https://github.com/amatray/claude-skills-public/raw/main/skills-pack/skills_pack.zip`
2. Unzip it: on Windows, right-click the file and choose **Extract All**
3. Move the unzipped `skills_pack` folder inside the project folder where your workshop skills are installed
4. In Visual Studio Code: **File**, **Open Folder**, choose that **project folder**, not `skills_pack`
5. Open `skills_pack/README.md` in the file list, then open the Claude panel

The slides give the README section and line number for each prompt

Outline

What a skill is

How to get Claude to do what you want

Four choices in a skill's text

When to call an agent

The literature skill, read line by line

Change a skill you already have

Build a small skill, then improve it

A skill is a folder with one text file

Inside `.claude/skills/`

In the folder	What it is
`SKILL.md`	Instructions. The only required file
`scripts/`	Code the skill runs
`references/`	Material it reads when it needs it
`templates/`	Output format the skill uses
`gotchas.md`	What has gone wrong before

SKILL.md starts with YAML, followed by Markdown instructions

Minimal example: `voice-learn-idb/SKILL.md`

```
---
name: voice-learn-idb
description: >-
  Compare your draft with the user's corrected version
  and record agreed writing preferences in corrections.md.
  Use when the user says "learn from my edits"
  or shares a corrected version of your draft.
  Never edit a writing guide.
---

# Learn from my edits
1. Find your draft and the version I sent.
2. Ask why I made each recurring change.
3. Record it in corrections.md; ask for my yes.
```

The description tells Claude what the skill does and when to open its instructions

SKILL.md starts with YAML, followed by Markdown instructions

Minimal example: `voice-learn-idb/SKILL.md`

```
---
name: voice-learn-idb
description: >-
  Compare your draft with the user's corrected version
  and record agreed writing preferences in corrections.md.
  Use when the user says "learn from my edits"
  or shares a corrected version of your draft.
  Never edit a writing guide.
---

# Learn from my edits
1. Find your draft and the version I sent.
2. Ask why I made each recurring change.
3. Record it in corrections.md; ask for my yes.
```

The description tells Claude what the skill does and when to open its instructions

Trigger phrases: the highlighted lines say when to select the skill

Outline

What a skill is

How to get Claude to do what you want

Four choices in a skill's text

When to call an agent

The literature skill, read line by line

Change a skill you already have

Build a small skill, then improve it

How to give Claude instructions

Written instructions

AI reads and **interprets** what you want

Example

> Check that every firm-year appears only once before estimating

How to give Claude instructions

Written instructions

AI reads and **interprets** what you want

Example

> Check that every firm-year appears only once before estimating

Executable code

Code → purely deterministic

Same requirement in Stata

```
isid firm_id year
```

Stata checks uniqueness and stops if it finds duplicates

In the skill: Run the uniqueness check before estimating

If it fails, stop and ask the researcher how to resolve the duplicates

Use scripts for deterministic operations; AI handles interpretation

Scripts: explicit rules with a checkable result

Operation	What the script does
Authentication	Verify the required account is authenticated; stop otherwise
Required steps	Check that validation passed for the current inputs before producing the report
Data rules	Check that firm-year pairs are unique; stop if duplicates exist
Calculations	Count working days using an explicit calendar rule

Use scripts for deterministic operations; AI handles interpretation

Scripts: explicit rules with a checkable result

Operation	What the script does
Authentication	Verify the required account is authenticated; stop otherwise
Required steps	Check that validation passed for the current inputs before producing the report
Data rules	Check that firm-year pairs are unique; stop if duplicates exist
Calculations	Count working days using an explicit calendar rule

AI: interpretation and judgment

Is this paper relevant to my research? What do these results mean?

Outline

What a skill is

How to get Claude to do what you want

Four choices in a skill's text

When to call an agent

The literature skill, read line by line

Change a skill you already have

Build a small skill, then improve it

Split the work into phases and specify where to stop

Phases / Arguments / Output format / Missing information

Specify how the skill proceeds, its arguments, its output format, and what to do when information is missing

In `SKILL.md` (example)

```
## Phase 1: Read
Read the files and ask for missing information.

## Phase 2: Analyze
Present your findings and identify gaps.
STOP. Discuss the findings with the user.

## Phase 3: Propose
Present a plan.
Wait for approval before continuing.

## Phase 4: Execute
Carry out the approved plan.
```

Use arguments to control how the skill runs

Arguments are the text you type after the skill name

In `SKILL.md` (example)

```
## Arguments
Read $ARGUMENTS for these options:
- quick: produce abbreviated output.
- depth:light/standard/deep: choose the thoroughness.
- file:path/to/file: use this input file.
- focus:dimension: focus on this aspect.
- dryrun: show the plan without executing it.
- help: explain the options and stop.
```

Example invocation

```
/my-skill depth:deep file:report.md
```

Give the skill the Markdown structure of its output

In `SKILL.md` (example)

```
## Output
Save one file: PROJECT_STATUS.md.
Use this Markdown structure:

# [Deliverable] | Due: [date]

## Next actions
- [Action] | Due: [date]

## Waiting on others
- [Who] | [What they owe] | Due: [date]

## Done
- [Completed item]
```

Give the skill the Markdown structure of its output

In `SKILL.md` (example)

```
## Output
Save one file: PROJECT_STATUS.md.
Use this Markdown structure:

# [Deliverable] | Due: [date]

## Next actions
- [Action] | Due: [date]

## Waiting on others
- [Who] | [What they owe] | Due: [date]

## Done
- [Completed item]

If a date is missing, write "no date".
```

When something is missing, should the skill carry on or stop?

Write the gap and carry on

- If a due date is missing, write no `date` and continue
- When a search finds nothing, the literature skill writes "no country-specific evidence found on X" and goes to the next topic

Stop and ask

- Literature skill with no topic: no search runs, and it asks its one question
- If the project folder is not specified, stop and ask which folder to use

When something is missing, should the skill carry on or stop?

Write the gap and carry on

- If a due date is missing, write no `date` and continue
- When a search finds nothing, the literature skill writes "no country-specific evidence found on X" and goes to the next topic

Stop and ask

- Literature skill with no topic: no search runs, and it asks its one question
- If the project folder is not specified, stop and ask which folder to use

⚠ Remember, the **predictive monkey** naturally fills gaps with something **plausible**. Make sure it writes gaps down and does not fill in missing information

Outline

What a skill is

How to get Claude to do what you want

Four choices in a skill's text

When to call an agent

The literature skill, read line by line

Change a skill you already have

Build a small skill, then improve it

An agent is a separate Claude session with a short brief

Skill and agent

A skill runs in your conversation and can ask you questions

An agent starts fresh, sees only its brief, and returns one message

An agent is a separate Claude session with a short brief

Skill and agent

A skill runs in your conversation and can ask you questions

An agent starts fresh, sees only its brief, and returns one message

Four reasons to call one

Parallel work: independent pieces run at the same time

A lighter Claude model: cheaper and faster for the bulk reading. Nothing leaves the tool the Bank approved

A clean conversation: the agent searches and reads, and only its result comes back

Fresh evaluation: a reader who did not write the draft looks for what is wrong with it

An agent is a separate Claude session with a short brief

Skill and agent

A skill runs in your conversation and can ask you questions

An agent starts fresh, sees only its brief, and returns one message

Four reasons to call one

Parallel work: independent pieces run at the same time

A lighter Claude model: cheaper and faster for the bulk reading. Nothing leaves the tool the Bank approved

A clean conversation: the agent searches and reads, and only its result comes back

Fresh evaluation: a reader who did not write the draft looks for what is wrong with it

Asked “is this good?”, Claude tends to say yes, most of all about a draft it wrote itself

The brief is all the agent knows

Rule for the brief	Why
Say everything	Agent cannot ask you a question and has not seen your conversation

The brief is all the agent knows

Rule for the brief	Why
Say everything	Agent cannot ask you a question and has not seen your conversation
Give the file, with its path	Naming a reviewer (“a skeptic”) does not mean it has read what you have in mind

The brief is all the agent knows

Rule for the brief	Why
Say everything	Agent cannot ask you a question and has not seen your conversation
Give the file, with its path	Naming a reviewer (“a skeptic”) does not mean it has read what you have in mind
Leave your conclusion out	A reviewer told what you think tends to hand it back confirmed

Limit: Each agent uses more of your allowance, and you never see what it left out

The brief is all the agent knows

Rule for the brief	Why
Say everything	Agent cannot ask you a question and has not seen your conversation
Give the file, with its path	Naming a reviewer (“a skeptic”) does not mean it has read what you have in mind
Leave your conclusion out	A reviewer told what you think tends to hand it back confirmed

Limit: Each agent uses more of your allowance, and you never see what it left out

Key takeaway

Most small skills need no agent, and the one you build today has none

Outline

What a skill is

How to get Claude to do what you want

Four choices in a skill's text

When to call an agent

The literature skill, read line by line

Change a skill you already have

Build a small skill, then improve it

Read the literature skill from its trigger to its final report

Open `.claude/skills/literature-idb/SKILL.md`

This is the map of that file; each row points to a part we will examine

Field or heading in the file	Decision it controls
`description:`	When Claude selects this skill
The ladder	Where to search, and in what order
Country and scope, before any search	What to ask before searching
Write your section of the manifest	How to record each source
Done means the checker says so	What must pass before completion
The list is a bibliography; now write the review	How to synthesize the evidence
Report the evidence, not the checker	What to tell the user
Failures	Which mistakes the instructions guard against

Before searching, the skill checks the country and topic

```
/literature-idb chile fiscal policy
```

Country: Chile Topic: fiscal policy

- Country: from what you typed, else from `config/country.md`, else it asks
- Topic: from the rest of what you typed. If nothing is left, it stops and asks, in these exact words

> What should I search the literature for? Name a topic, for example fiscal policy, or labor informality. Or say **everything**, and I will collect everything written about the country I can find.

From the skill file: "Never fill a missing scope yourself"

The search order inside the country: institutions first, journals second

1. **Institutional and grey literature:** IDB publications, IMF working papers and Article IV selected issues, World Bank, CEPAL, CAF, central bank and ministry working papers, rating-agency actions
2. **Then academic databases**

The skill's own reasons

"Institutional and grey literature first, because for these countries that is where the evidence lives"

"A reference copied from another paper's bibliography is a lead to chase, not an entry"

When the country runs out: widen, and label how far you went

Next step of the order	Label and reason, from the skill file
Regional study with this country in the sample	"Labeled as such, because a regional average is not a country fact"
A named comparator country	An analogy: the chapter "may say 'in Peru, X', never 'here, X'"
Nothing found	"no country-specific evidence found on <topic>", with where it looked. "That line is a finding. Do not fill the void"

Every entry carries the tier it came from, and a topic with no evidence gets its own line

The parent skill moved the same search order from text into a script

`literature-idb`

Academic step: three words of text, “Then academic databases”

Which ones, in which order: left to Claude on each run

A ladder that lives in prose is a ladder the model can skip a rung of without noticing

The parent skill moved the same search order from text into a script

lit-review

```
lit-review/
|-- SKILL.md
|-- gotchas.md
|-- references/
`-- scripts/
   `-- search_ladder.py
```

Excerpt from SKILL.md

Every search in this skill starts by running the ladder script.

<this skill> below is the folder this SKILL.md sits in.

```
python3 <this skill>/scripts/search_ladder.py \
  run "<topic>" \
  --query "<query 1>" --query "<query 2>" \
  --n 10 --out output/_intermediate
```

The literature skill records sources in a table for the checker

`manifest/literature-idb.md`

#	Citation	Link	What it shows (one line)	Tier

The link takes one of three forms

1. **A web address:** the script downloads the document
2. **A file path:** a public document a person saved by hand, because the site refuses scripts
3. **“not attached:”** and where it is recorded, for material that must not be redistributed

A script cannot read a paragraph, so the skill asks for a table in a fixed shape

What the script does to every row

```
python3 check_literature.py --root .
```

1. Reads the list, row by row
2. Downloads each web address; if the site refuses, tries again under two other identities
3. Records a fingerprint of each downloaded file, so it can be recognised a year later
4. Reads the document and looks for the words of the citation: under half found, the row fails
5. Fails any link with a tracking tag such as `utm_source`, the mark of a citation pasted from a chat
6. Checks a file saved by hand the same way, and checks that "not attached" says where
7. Writes a log of what happened to each row

The Python code ensures that the check is systematic rather than relying on Claude's interpretation

The skill can use AI to act on the script's deterministic output

1. Example: a broken download link

1. Claude lists a World Bank report with a download link
2. The script tries to download it and reports that the link does not work
3. Claude searches for a working link to **the same report**, then reruns the checker

Claude must not change the citation just to make the check pass

Warning

2. Passing does not prove the source is correct

A wrong edition passed because the publisher, country and series name matched enough of the citation's words. You still need to check the edition

Record mistakes in gotchas.md so the skill reads them next time

Example: Claude starts searching before you specify a topic

In `gotchas.md`

```
## Searching without a topic
Claude inferred a topic from the project notes and started searching.

Ask the user for the topic instead. Wait for the answer
before running any search.
```

In `SKILL.md`

```
Before starting, read gotchas.md and follow its instructions.
```

Recording the mistake is not enough: **the skill must be told to read the file**

Outline

What a skill is

How to get Claude to do what you want

Four choices in a skill's text

When to call an agent

The literature skill, read line by line

Change a skill you already have

Build a small skill, then improve it

Three depths of change, from cheapest to most invasive

Where it is decided	What you change	Example
A file the skill reads	File in your project; the skill is untouched	`reminder_style.md`: "This file is yours"
The text of the skill	Words in `SKILL.md`	Search order of the literature skill
A script	Code inside the skill	`check_literature.py`

When you do not know which one, ask first

`README.md`, section B3, prompt starts at line 116

> Read the skill [name] and tell me where this behavior is decided: [the behavior]. Is it a file the skill reads, the text of the skill, or a script? Then propose the smallest change. Change nothing yet.

Three depths of change, from cheapest to most invasive

Where it is decided	What you change	Example
A file the skill reads	File in your project; the skill is untouched	`reminder_style.md`: "This file is yours"
The text of the skill	Words in `SKILL.md`	Search order of the literature skill
A script	Code inside the skill	`check_literature.py`

When you do not know which one, ask first

`README.md`, section B3, prompt starts at line 116

> Read the skill [name] and tell me where this behavior is decided: [the behavior]. Is it a file the skill reads, the text of the skill, or a script? Then propose the smallest change. Change nothing yet.

⚠ If you add instructions to a skill, keep `SKILL.md` under 500 lines and about 5,000 tokens. Put detailed examples and reference material in `references/`

Your turn: add your own source to the search order

Already in the list: IDB, IMF, World Bank, CEPAL, CAF, central banks and ministries, rating agencies

`README.md`, section B1, prompt starts at line 86

> I want to change the literature-idb skill. Do not run the skill. First tell me which copy of it runs on this laptop and where it is. Look only in the skills folder inside the .claude folder of my home folder and of this project; do not search the rest of the laptop. If you find more than one copy, list them and ask me which one to edit. In the ladder of that copy, add [a source that is not in the list, for example: OECD Economic Surveys] to Tier 1, at the end of the list of institutions, just before the sentence "Then academic databases". Change nothing else. Show me the lines before and after, and wait for my yes before saving.

- Fill the blank with your own source, paste, read the line before and after, then type yes
- Claude asks permission to write in the `.claude` folder, perhaps more than once: allow it each time

Check the change in a new session, by reading the file

1. Start a new conversation: the speech bubble with a plus sign, top right of the Claude panel
2. Paste this question

`README.md`, section B2, prompt starts at line 104

> Read the literature-idb skill that runs on this laptop and list the Tier 1 sources in order. Look only in the skills folder inside the .claude folder of my home folder and of this project; do not search the rest of the laptop. Do not run the skill.

Your source is last among the institutions, and nothing else moved

Outline

What a skill is

How to get Claude to do what you want

Four choices in a skill's text

When to call an agent

The literature skill, read line by line

Change a skill you already have

Build a small skill, then improve it

The project index and the status page have different jobs

```
My project/
├── PROJECT_INDEX.md      Where files live; which versions are current
├── PROJECT_STATUS.md     Current status and next actions
├── AGENTS.md / CLAUDE.md Standing instructions for your AI tools
├── research/             Background research and source notes
├── drafts/               Work in progress
└── transcripts/          Project meeting transcripts
```

- **Create the index separately:** ask Claude to create `PROJECT_INDEX.md` during project setup, listing where files live and linking to the status page
- **In this exercise:** `project-status-idb` creates and updates only `PROJECT_STATUS.md`: the deliverable, next actions, what you are waiting for, and what is done
- **The status skill does not create the index** or require one to run

Create the project index before the status-skill exercise

Paste this request in your project folder:

`README.md`, section C0, prompt starts at line 140

> Draft PROJECT_INDEX.md for the project folder I have open. List the main files and folders, what each contains, and which versions are current. Mark anything uncertain rather than guessing.
>
> Link to PROJECT_STATUS.md if it exists; otherwise mark it as not yet created. Do not move or rename any files. If an index already exists, propose changes to it instead.
>
> Show me the proposed index and wait for my yes before writing it.

This creates the index directly. The next exercise builds a skill to maintain the status page

Next session: skills to **clean an existing folder** and correctly set up a **new project folder**

You build the skill by writing one request to Claude

README.md, section C1, prompt starts at line 160

Create a skill called project-status-idb. Save it as project-status-idb/SKILL.md in the skills folder inside the .claude folder of my home folder. Everything below describes what the skill does when I use it later. Do not start doing it now: only write the skill file.

Job: it keeps one status page for the project folder I have open, so I can see what I must do next and what I am waiting for from others.

When to use it: when I say "[your phrase to set up or update the page]" or "[your phrase to ask what is late]".

What to ask: the first time in a project, ask me three questions, one at a time, typed in the chat, never as a multiple-choice box: what the deliverable is and its date; what my next actions are; who owes me what and by when. Never fill in an answer I did not give.

Steps: ask, then show me the page you propose, then stop and wait for my yes, then write it. If PROJECT_STATUS.md already exists, never start over: read it, ask me what changed, show me the changes, and wait for my yes. When I only ask what is late, read the page and answer; change nothing. For each item, say how many days late it is, or "not due yet".

Output: one file, PROJECT_STATUS.md, in the project folder. First line: the deliverable and its date. Then three sections in this order: Next actions; Waiting on others (who, what, due date); Done.

When something is missing: if no folder is open, or you are not sure which project I mean, stop and ask; do not search for a folder. If a due date is missing, write "no date" and carry on. [your own line: one more thing the skill needs, and whether it carries on or stops without it]

When you are done, show me the full text of the skill file.

Three blanks are yours: two phrases, and one line for a missing piece. Then paste, and allow the permission

Read what Claude wrote and find your sentences in it

description: Use when the user says “set up my status page” or “what is late on my project” [...]

Never write an answer the user did not give. Not a date, not an owner, not a next action, not a deliverable.

If no folder is open, stop and ask which project folder to use. [...] Do **not** search the filesystem for a candidate folder

One question at a time. Ask the first, wait for the answer, then ask the second.

Stop and wait for the user’s yes.

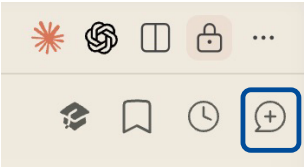
Look for

1. Your two phrases, in the description
2. The stop before writing
3. Your line for the missing piece

For each of your sentences: is it there, and did it keep its meaning?

Use it in a new session, in your project folder

1. Start a new conversation: click the highlighted button



2. Type your first phrase. If nothing happens, type `/project-status-idb`
3. Give one item whose date has already passed

Check the first run in a folder with no status page

Claude asks about the deliverable and date, your next actions, then who owes you what and when

It waits for each answer, shows a proposed page, and asks permission to save

After your yes: open `PROJECT_STATUS.md` and check the headings Next actions, Waiting on others, and Done

Tell the skill how to count days late

If you do not specify the counting rule, **Claude may choose one without telling you**

Table: An item was due Friday; today is Monday

Counting rule	Answer
Calendar days	3 days late
Working days	1 working day late

Notes: Exclude the due date and include today. Working days exclude Saturday and Sunday.

Claude could report **3 days late** or **1 working day late**, depending on the rule it chose

Make the counting rule explicit. Neither definition is inherently the correct one

Ask Claude to put that rule in a script

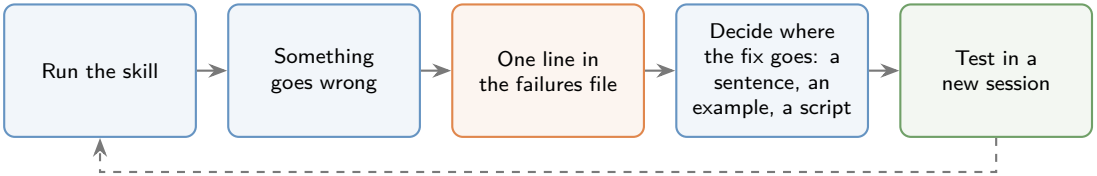
Paste this request:

`README.md`, section D2, prompt starts at line 290

> Update project-status-idb to report working days late.
>
> Count Monday to Friday, including weekday public holidays. Exclude the due date and include today.
>
> 1. Add workdays.py and require the skill to run it whenever it calculates lateness.
> 2. Test it:
>   6 November 2026 to 9 November 2026: 1 working day.
>   3 November 2026 to 24 November 2026: 15 working days.
> 3. Record the calendar-day mistake in gotchas.md. Tell the skill to read that file before running.
>
> Show me the code, test results and proposed SKILL.md edits. Wait for my approval before installing the changes.

You check the rule and test results before approving the change

The loop is the same one as for your voice



The same request works for any skill

`README.md`, section D4, prompt starts at line 333

> The skill [name] did [what happened]. I wanted [what should happen]. Read the skill and tell me where this is decided: a file it reads, its text, or a script. Propose the smallest change, and add one line to its gotchas.md naming the kind of failure. If the skill has no gotchas.md, create it and add a line to the skill telling it to read that file before it runs. Show me before and after, and wait for my yes.

Possible extensions, at home

- **Propose "Waiting on others" entries from email:** Marie writes "I'll send the figures by Friday"; the skill proposes `Marie | Figures | Due Friday`, and you approve before it saves. The Bank must allow the email connection first
- **Publish the page where your team can open it:** same condition

For your own skill: `README.md`, section C6, prompt starts at line 245

Each one adds a dependency, and brings back the question: carry on or stop?

Key takeaway

You write the request and check the skill Claude writes from it

Thank you!

Adrien Matray adrien.matray@gmail.com

Appendix: gates, tools and depth

Topic	What it is
Confirmation gates, three tiers	Apply automatically, confirm in one batch, or confirm item by item, matched to the risk of the action
Pre-approved tools	Stating up front which tools the skill may call, to avoid permission prompts
Depth levels and auto-detection	Light, standard and deep runs; behavior inferred from the content. The literature skill refuses to infer its topic

Appendix: state, size and a checklist before sharing

Topic	What it is
Config and state	Files the skill only reads, against files it reads and writes
Size and placement	Under five hundred lines and about five thousand tokens; detailed examples and reference material in `references/`
Blattman's checklist	Works with no arguments, output format specified, errors handled, does one thing, arguments documented, tested on real tasks