WORKSHOP OVERVIEW

Skills in Claude Code

Build expert skills conversationally — describe what you want, the skill builder writes the file. No code, no terminal required.

SECTION 1

What Is a Skill?

A SKILL.md file · Triggered by description · Loaded automatically

A skill is a plain Markdown file called `SKILL.md` stored in a folder. Before Claude starts a task, it scans skill descriptions to decide if one is relevant — then reads the full file to get expert instructions.

① FRONTMATTER

description

Tells Claude **when** to apply the skill. Specific verbs, clear scope. The only part Claude reads when deciding to load.

② MARKDOWN BODY

instructions

Tells Claude **how** to do the task well. Tone rules, structure, examples — everything you'd put in a style guide.

💡 **Mental model**

Hand Claude a job description before it starts work. The description field is the trigger logic. The rest of the file is the expert guidance. Write the description with specific verbs and a clear scope.

SECTION 2

How Claude Decides to Use a Skill

A four-step automatic process · Runs on every task

STEP 1

User Prompt

"Write an email to a client…"

→

STEP 2

Scan Descriptions

Reads description fields from all enabled skills.

→

STEP 3

Load SKILL.md

Reads the full file for best practices.

→

STEP 4

Execute Task

Applies your expert guidance.

SECTION 3

Anatomy of a SKILL.md

YAML frontmatter · Markdown body · Optional examples

```
---
name: email-writer
description: "Use this skill whenever the user asks to write,
  draft, or compose a professional email, follow up, or
  reply to an email."
---

# Email Writer Skill

## Tone & Voice
- Clear, concise, professional
- Avoid jargon and filler phrases

## Structure
1. Subject line (specific + action)
2. Opening — one sentence, no filler
3. Body — max 4 short paragraphs
4. ONE clear ask at the end
5. Sign-off: "Best,"

## Example
**Subject:** Project update — new date: March 15

Hi Sarah,

We're running one week behind on the dashboard redesign
due to an API delay. New delivery date is March 15.

Could you confirm this works for your team?
```

```
Best,
[Your name]
```

FIELD	WHAT IT DOES
name	Short identifier. Appears in logs when Claude loads the skill.
description	**This triggers the skill.** Claude reads only this field to decide whether to load the rest. Specific verbs, clear scope.
Instructions	Tone, rules, structure — the expert guidance. Use headers, bullets, numbered lists.
Example *(optional)*	Optional but powerful — one concrete example beats three paragraphs of rules.

SECTION 4 · SETUP

Enable the Skill Builder

Toggle skill-creator ON · Verify with /skills

The **skill-creator** is a built-in Anthropic skill that turns Claude into an interactive skill designer. Enable it once and you're ready to build any skill conversationally — no files or folders to manage.

4.1 Open Customize > Skills in the desktop app

⏱ 1 min

1. Click the **gear icon** or open **Customize** in the left sidebar.
2. Select **Skills** from the submenu.
3. You'll see a list of available skills — built-in Anthropic skills are shown at the top.

4.2 Toggle `skill-creator` on

⏱ 1 min

Locate **skill-creator** in the Anthropic skills section. Click its toggle — it turns teal when active. The skill is now available in every chat window.

> 💡 **While you're here**
>
> Also enable `docx`, `pptx`, `xlsx`, and `pdf` if they're not already active — you'll use them in Stage 2.

4.3 Verify it's active

⏱ 1 min

Open a new chat window in Claude Code and type:

```
/skills
```

You should see `skill-creator` listed as active.

WORKSHOP STAGES

The Path

Basic to advanced · Each stage uses the skill builder

STAGE 1 · BASIC

Email Writer

Build your first skill: professional client emails with specific subject lines, no filler opener, one clear ask. **~10 min**

STAGE 2 · BUILT-IN

Document Skills

Enable built-in `docx`, `pptx`, and `pdf` skills. Live demo comparing output with and without each. **~10 min**

STAGE 3 · CUSTOM

Build Any Skill

The general five-step loop: pick a task, invoke the builder, refine, confirm, test. Works for any repeating task. **~10 min**

STAGE 3B · INTERMEDIATE

Sharpen Your Skill

Tighten the trigger, add negative rules and conditionals, run a deliberate eval. Four upgrades. **~15 min**

STAGE 3C · REAL WORLD

Contract Reviewer

Build a skill that reads any contract, flags weak or risky clauses, and suggests plain-English fixes. **~15 min**

STAGE 3D · HAND-EDIT

Edit SKILL.md by Hand

Open the raw `SKILL.md` file, tweak by hand, and sideload it into the desktop app via Settings → Capabilities → Skills. **~10 min**

STAGE 4 · ADVANCED

Game-Making Skill

Generate a full `retro-shooter` expert skill — canvas, delta-time loop, chiptune, waves — then build a game with it. **~15 min**

STAGE 1 · BASIC

Email Writer — Your First Skill

One prompt to the skill builder is all it takes

1.1 Invoke the skill builder

⏱ 2 min

Open a new chat in Claude Code and paste this prompt:

PROMPT — INVOKE THE SKILL BUILDER

"Use the skill builder to create an email-writer skill. It should write professional client emails: specific subject line, no filler opener, max 200 words, one clear ask at the end, sign-off with 'Best,'"

1.2 Answer clarifying questions

⏱ 2 min

The builder will ask about preferred tone, sign-off, word limits, and special rules (e.g. always CC a manager on client emails). Answer in plain language.

1.3 Review the generated SKILL.md

⏱ 2 min

The builder shows you the complete file — frontmatter, rules, and an example. Ask for any changes in plain English:

- "Change the sign-off to 'Kind regards,'"
- "Add a rule about keeping emails under 150 words"
- "Always CC the account manager on client emails"

1.4 Confirm and save

⏱ 1 min

When you're happy, type **save** or **confirm**. The builder writes the `SKILL.md` to your skills folder automatically — no file manager needed.

1.5 Test it — see the difference

⏱ 3 min

Start a fresh chat. Claude should mention reading the email-writer skill before responding:

```
"Write an email to a client about a project delay."
"Draft a follow-up to a colleague who hasn't responded in 3 days."
"Reply to a customer complaint about a late delivery."
```

Look for: specific subject line · no "Hope this email finds you well" · one concrete ask · your custom sign-off.

STAGE 2 · BUILT-IN

Document Skills

Enable docx, pptx, xlsx, pdf · Live before/after demos

Anthropic ships ready-made skills that turn Claude into an expert with each format. Enable them in **Customize > Skills**, then run the prompts below in two fresh chats — once with the skill off, once with it on — and watch the output sharpen.

DOCX

Word documents

Generates real `.docx` files with proper headings, tables, and styles instead of plain Markdown.

PPTX

PowerPoint slides

Creates designed slide decks — slide layouts, speaker notes, real `.pptx` output.

XLSX

Excel spreadsheets

Builds working spreadsheets with formulas, formatting, and multiple sheets.

PDF

PDF reading

Lets Claude read uploaded PDFs (contracts, papers, scanned docs) end-to-end.

DEMO PROMPT

"Create a one-page Word doc summarising the four prompting patterns from the basic workshop. Include a table of patterns with examples, a short intro, and a heading hierarchy."

Run this twice: once with `docx` off (you'll get markdown), once with it on (you'll get a real downloadable Word file).

STAGE 3 · CUSTOM

Build Any Skill — The Five-Step Loop

Pick a task · Invoke · Refine · Confirm · Test

STEP	WHAT YOU DO
1. Pick a task	Anything you've asked Claude to do more than twice — meeting notes, code review, ticket writing, status updates.
2. Invoke	"Use the skill builder to create a [name] skill that [description]."
3. Refine	Answer clarifying questions and request edits in plain English.
4. Confirm	Type `save`. The builder writes the file to your skills folder.
5. Test	Start a fresh chat, run a trigger phrase, check Claude's thinking output for the skill name.

STAGE 3B · INTERMEDIATE

Sharpen Your Skill

Four upgrades that make any skill measurably better

Revisit the email-writer skill from Stage 1. Four targeted upgrades — all in plain language, no files to edit manually.

3b.1 Trigger Engineering

Tighten the description so it fires exactly when you want

The description field is the only part Claude reads when deciding whether to load a skill. Vague descriptions over-fire or under-fire.

TOO VAGUE

```
Use this skill when the user wants to
communicate with someone.
```

PRECISE

```
Use this skill whenever the user asks to
write, draft, or compose a professional
email, follow-up, or reply.
```

PROMPT — TIGHTEN THE TRIGGER

"Use the skill builder to edit my email-writer skill. Tighten the description so it triggers on: write an email, draft an email, compose a message, follow up, reply to an email. It should NOT trigger when I ask to summarise an email or translate one."

3b.2 Negative Examples

Tell Claude what to never do

One concrete "never" beats a paragraph of positive instructions.

🚫 **Effective negative rules**

- Never open with "Hope this email finds you well" or any filler greeting.
- Never use more than one exclamation mark per email.
- Never include more than one ask per email.

3b.3 Conditional Instructions

Different rules for different situations

Skills can branch. Write plain-English conditions and Claude applies the right rule based on context.

🔀 Examples of conditions

- If the email is to an internal colleague, use a casual tone and skip the formal sign-off.
- If the email is a complaint response, open by acknowledging the issue before explaining.
- If the user provides a deadline, always include it in the subject line.

3b.4 Skill Evaluation

Verify it fires when it should and stays quiet when it shouldn't

A good skill eval is six prompts: three that should trigger the skill, three that should not. Run them in a fresh chat each time so there's no conversation carry-over.

SHOULD TRIGGER

```
"Write an email to a client about a delay."
"Draft a follow-up to someone who hasn't
replied."
"Reply to a customer complaint."
```

SHOULD NOT TRIGGER

```
"Summarise this email thread."
"Translate this email to Spanish."
"What does this email mean?"
```

💡 How to check if the skill fired

Look at Claude's thinking output at the top of the response — it names the skill if it loaded it. No skill name = skill did not fire.

STAGE 3C · REAL WORLD

Contract Reviewer

A skill that flags risky clauses and suggests plain-English fixes

📎 **Before you start — enable the PDF skill**

Go to **Customize > Skills** and toggle `pdf` ON. This lets Claude read contract files you upload directly — no copy-pasting required.

3c.1 Build the skill with one prompt

⏱ 3 min

PROMPT — INVOKE THE SKILL BUILDER

"Use the skill builder to create a contract-reviewer skill. When I paste or upload a contract, it should: read every clause carefully, flag weak or risky points, explain why each one is a problem in plain English, and suggest a specific fix. Organise the output as a numbered list of findings. Each finding should have: the original clause or phrase, what the risk is, and a suggested rewrite."

3c.2 Tell it what weak points to look for

⏱ 4 min

VAGUE LANGUAGE

Words like "reasonable", "timely", "as soon as possible" with no defined meaning. Either party can interpret them differently.

ONE-SIDED TERMINATION RIGHTS

Only one party can terminate, or notice periods are very short.

UNCAPPED LIABILITY

No limit on how much one party can owe the other if something goes wrong.

MISSING DELIVERABLES OR DEADLINES

Work described in general terms with no specific output, date, or measurable acceptance.

AUTOMATIC RENEWAL CLAUSES

Contract renews unless cancelled within a short window — easy to miss, costly to break.

IP OWNERSHIP AMBIGUITY

Who owns work produced under the contract is unclear or defaults to the wrong party.

3c.3 Set the output format

⏱ 2 min

PROMPT — DEFINE THE OUTPUT FORMAT

"Each finding should follow this format exactly — Issue: [one sentence naming the problem]. Risk: [why this matters, in plain English, one or two sentences]. Suggested fix: [a rewritten version of the clause or specific language to add]."

3c.4 Save and test with a real clause

⏱ 4 min

Type **save** to confirm. Open a fresh chat and paste:

```
"Review this contract clause:

The vendor shall deliver the project within a reasonable timeframe.
Payment is due upon satisfactory completion. Either party may
terminate this agreement with 3 days written notice. The vendor
retains all intellectual property created under this agreement
unless otherwise agreed in writing."
```

A well-built skill should return at least four findings — vague deadline, undefined "satisfactory", short termination notice, and IP ownership risk.

📄 **Works with uploaded PDFs too**

With the `pdf` skill enabled, you can upload a full contract file instead of pasting text. Claude reads the whole document and applies the contract-reviewer skill to every clause automatically.

STAGE 3D · HAND-EDIT

Edit SKILL.md by Hand & Load It in the Desktop App

Open the file · Tweak it · Sideload via Settings → Capabilities → Skills

The skill builder is fast, but every skill is just a Markdown file on disk. Open it, change a line, save, then sideload the folder into the Claude Desktop app — no builder required. This is the underlying mechanic the builder is automating, and it's the only way to wire in resources Claude can't write for you (CSVs, examples, reference scripts).

> 📁 **Where the file lives**
>
> Skills built in earlier stages live under `~/.claude/skills/<skill-name>/SKILL.md` on macOS/Linux, or `%USERPROFILE%\.claude\skills\<skill-name>\SKILL.md` on Windows. Each skill is its own folder — drop extra files (templates, reference data) right next to `SKILL.md`.

3d.1

Open the existing skill folder

⏱ 1 min

Pick the `email-writer` skill from Stage 1. Open its folder in your file manager or favourite editor (VS Code, Sublime, even TextEdit/Notepad).

macOS / Linux

```
cd ~/.claude/skills/email-writer
ls
# SKILL.md
code SKILL.md   # or: open -e SKILL.md
```

Windows

```
cd %USERPROFILE%\.claude\skills\email-writer
dir
notepad SKILL.md
# or open the folder in Explorer
```

3d.2

Read the two parts: frontmatter & body

⏱ 1 min

The top YAML block between the `---` fences is the **frontmatter** — Claude reads *only* the description here when deciding whether to load the skill. Everything below is the **body** — the rules Claude follows once the skill is active.

```
---
name: email-writer
description: Use this skill whenever the user asks to write,
  draft, compose, or rewrite a professional email. Triggers
  include 'write an email', 'draft an email', 'reply to'.
---

# Email Writer

## Rules
- Subject lines must be specific (no "Quick question").
- Never open with "Hope this email finds you well".
- One clear ask per email.
- Sign off with "— Doron".
```

3d.3 Make a deliberate manual edit

⏱ 2 min

Change something the builder would have to be coaxed into. Three good practice edits:

> ✏️ **Try one of these**
>
> - Add a new bullet under **Rules**: "Match the language of the incoming email — if it's in Hebrew, reply in Hebrew."
> - Add a new section ## **Examples** with one good and one bad email side by side.
> - Tighten the description by adding negative triggers: "Do NOT trigger when the user asks to summarise or translate an existing email."

Save the file. That's it — no compile step, no restart yet.

3d.4 Load it into the Claude Desktop app

⏱ 2 min

The desktop app reads skills from your local skills folder, but you have to point it there once. The flow is identical on macOS and Windows.

1. Open the **Claude Desktop** app.

2. Click your profile / avatar → **Settings**.
3. Go to **Capabilities** → **Skills** (or **Customize** → **Skills** in newer builds).
4. Click **Add skill** / **Import from folder** and pick the folder containing your edited `SKILL.md`.
5. Toggle the new `email-writer` entry **on**.

> 💡 **If the app already had this skill loaded**
>
> Toggle it off, then on again — the desktop app re-reads the file on each enable. Some builds require a full quit (`⌘Q` / right-click tray → Quit) before edits are picked up.

3d.5 Verify the edit landed

⏱ 2 min

Open a fresh chat in the desktop app. Run a prompt that should fire the skill. Check Claude's thinking output at the top of the response — the skill name appears there when it loads.

PROMPT — VERIFY

"Write a short email to a client letting them know the report is delayed by two days."

> 🔎 **What you're looking for**
>
> Your manual edit should be visible in the output — the new rule applied, the new tone honoured, or the new section's example pattern reflected. If it's not, the file wasn't reloaded; toggle the skill off/on and try again.

3d.6 When to hand-edit vs. use the builder

A quick decision guide

HAND-EDIT

- One-line tweaks (description, a single rule)
- Adding reference files alongside SKILL.md
- Version-controlling skills in git
- Sharing a skill folder with a teammate

USE THE BUILDER

- Starting from scratch
- Restructuring large sections
- Adding negative examples or evals
- Anywhere you'd rather describe than write

STAGE 4 · ADVANCED

Game-Making Skill

Generate a richer technical skill than most people would write by hand

The skill builder can generate richer technical skills than most people would write by hand. Use it to create a full retro-shooter expert skill, then build a game with it immediately.

PROMPT 1 — BUILD THE RETRO-SHOOTER SKILL

"Use the skill builder to create a retro-shooter skill for building retro browser games as single HTML files. Cover: sub-genres (Space Invaders, Galaga, Asteroids, bullet hell), canvas 2D drawing, scrolling starfield, particle explosions, Web Audio API chiptune SFX, delta-time game loop, AABB collision detection, wave spawning, power-ups (rapid fire, spread shot, shield), HUD drawing, localStorage hi-score, and game states (title / playing / game over). Include a working code snippet for each major system. Write a description that triggers whenever anyone asks to build a game, shooter, or arcade project."

Refine it with follow-up prompts in the same conversation:

ADD CODE SNIPPETS

```
"Add code snippets for maybeDropPowerUp and the collectPowerUp handler."
```

TIGHTEN THE TRIGGER

```
"Make the description trigger on: game, shooter, arcade, Space Invaders, Galaga, Asteroids, bullet hell, canvas game, HTML game."
```

When satisfied, type **save**. Then test in a fresh chat:

PROMPT 2 — TEST THE SKILL

"Build me a simple Asteroids-style game — rotating ship, thrust, shooting rocks that split into smaller ones, wrap-around screen edges."

🎮 What to look for

Claude should mention reading the retro-shooter skill before starting. The output will be noticeably more polished: particle explosions, chiptune audio, delta-time movement, hi-score persistence, and proper game states — all from the skill.

QUICK REFERENCE

Skill Builder Cheatsheet

Everything you need in one place

TASK	HOW TO DO IT IN CLAUDE CODE
Enable the skill builder	Customize > Skills > toggle `skill-creator` ON
Create a new skill	"Use the skill builder to create a [name] skill that [description]"
Edit an existing skill	"Use the skill builder to edit my [name] skill — add [rule]"
List all active skills	`/skills` in any chat window
Disable a skill	Customize > Skills > toggle the skill OFF
Test a skill triggers	Fresh chat > type your trigger phrase > check thinking output for skill name
Hand-edit a skill	Open `~/.claude/skills/<name>/SKILL.md` in any text editor; sideload via Settings → Capabilities → Skills → Add skill
Export skill (for upload)	Right-click the skill folder > Compress / Send to ZIP
Upload skill to claude.ai	Settings > Customize > Skills > + > Upload a skill (ZIP)

Desktop App vs. claude.ai Web

ASPECT	CLAUDE CODE DESKTOP APP	CLAUDE.AI WEB
Create skills	Conversationally via skill-creator	Upload pre-built skills as ZIP
Edit skills	Ask the builder to update — saves in place	Must re-upload ZIP to update

ASPECT	CLAUDE CODE DESKTOP APP	CLAUDE.AI WEB
Where they work	All desktop app projects	Browser chat, Excel add-in, PowerPoint add-in
Manage	`/skills` to list; Customize to toggle	Customize > Skills toggle panel

IDEAS

Skills to Build Next

Any task you repeat is worth a skill

`meeting-notes`

Format and summarise meeting transcripts with a TL;DR and numbered action items with owners.

`code-review`

Enforce your team's PR review style — what to check, how to phrase feedback.

`release-notes`

Write changelogs in a consistent format from commit messages or diffs.

`ticket-writer`

Turn rough ideas into well-scoped Jira tickets with acceptance criteria.

`status-update`

Write weekly team status updates in a structured, scannable format.

`retro-shooter`

Expert skill for retro browser games — canvas, audio, game loop, waves, power-ups.

🔄 Skills are version-controllable

The `SKILL.md` files the builder creates live on disk. Commit them alongside your codebase and share them across your team — or export as ZIP and upload to claude.ai for browser and Office add-in use.

RECAP

Key Takeaways

Five rules to carry into your next workshop session

1. **Skills are plain Markdown files — no code required.** The skill builder writes the file; you just describe what you want in plain language.

2. **The skill builder makes creation conversational.** Enable `skill-creator` in Customize > Skills, describe your task, refine, then type *save*.

3. **The description field is what triggers a skill.** Specific verbs and clear scope — that's the only part Claude reads when deciding whether to load it.

4. **You can always hand-edit a SKILL.md and sideload the folder.** Use it for one-line tweaks, version control, and adding reference files the builder can't write.

5. **The same SKILL.md works everywhere.** Built in the desktop app, exported as ZIP, uploaded to claude.ai — write it once, use it in chat, Excel, and PowerPoint.
