# Clean up your Claude Code skills, one project at a time

If Claude Code feels like it got dumber, look at how many skills it reads on every turn. This guide sets each skill to one of four states for the project you are in, turns off idle plugins, and sets a lean default for everywhere else.

GraniteAI, Manchester, New Hampshire. Facts checked 23 September 2026 against Claude Code's docs and v2.1.280.
The guide page: graniteai.co/tools/clean-up-claude-code-skills

START HERE

# What you will have at the end

You use Claude Code with a pile of skills and plugins, and lately it misses things it used to catch. You have probably never opened /skills. Set aside one sitting per project. Claude drafts the cleanup; you check it and decide.

- [ ] Every skill in this project set to on, name-only, user-only or off, saved to the project's .claude/settings.local.json
- [ ] A lean global default in ~/.claude/settings.json, so a new project starts with less noise
- [ ] Plugins this project never uses turned off in /plugin
- [ ] One writing skill on where humanizer, copywriting and the-writer used to stack
- [ ] ponytail and humanizer installed, the other two fixes from the reel

### Why Claude Code seems to get dumber

## Every turn

Every skill that is on puts its name and description in front of Claude on every turn, used or not. Claude Code's docs: "Every skill in the skill listing adds to your context on every turn, whether or not Claude ever uses it." The full SKILL.md loads only when the skill is invoked.

## 1%

The skill listing gets 1% of the model's context window. When it overflows, Claude Code drops descriptions, starting with the skills you invoke least. The only warning goes to the debug log. A skill that loses its description gives Claude less to match your request against.

## Overlap

Skills that promise the same outcome compete for the same turn. The docs again: "If descriptions are vague or overlap, Claude may load the wrong skill or miss one that would help."

From Jason Rivard's own setup, with 252 skills installed: skills that clearly fit the job failed to fire, and three writing skills on at once made the writing worse. That is his experience, not a measured result. The mechanism above is what the docs describe.

### The four states

In /skills, every skill gets one of four states for the project you are in. The settings file stores the value in the second column.

| State | Saved as | What happens | Use it when |
|---|---|---|---|
| **on** | "on" | Claude sees the name and description. /name is in the menu. | Claude should reach for the skill on its own in this project. |
| **name-only** | "name-only" | Claude sees the name alone. /name is in the menu. The listing gets room back. | The name says what the skill does, and you want the space. |

| | | | |
|---|---|---|---|
| **user-only** | "user-invocable-only" | Claude does not see it. You can still type /name. | You want the skill only when you call it yourself. |
| **off** | "off" | Hidden from Claude and from the / menu. | This project never needs it. |

A skill with no entry counts as on. Turning a skill off deletes nothing: its files stay where they are.

STEP 1

# Open Claude Code in the project folder

When you close /skills, it saves your choices to **.claude/settings.local.json** in the folder where Claude Code started. Start it anywhere else and the choices land on the wrong folder. So move into the project first, then start Claude Code.

Run in a terminal

```
cd path/to/your-project
claude
```

**Your part:** swap in your project's path. Each project keeps its own file, so repeat the guide in every project you want lean.

STEP 2

# See what your skills cost

Look before you change anything. Type each of these into Claude Code as its own message.

Type into Claude Code

```
/context
```

```
/skill-doctor
```

- **/context** shows a Skills row with the size of the skill listing.
- **/skill-doctor** (v2.1.252 or later) shows what each skill costs and how often it gets used, and flags skills you have never invoked.
- **/doctor** estimates the listing cost, if your version predates /skill-doctor.

**Your part:** run both and read. The overflow warning goes only to the debug log, so the chat will not tell you when descriptions get dropped.

STEP 3

# Let Claude draft the cleanup

This prompt has Claude read the project, list your skills with their current states, and suggest a state for each one with a reason. It stops and changes nothing until you answer.

Paste into Claude Code

```
Audit my Claude Code skills for this project.
1. Read this project first: the README, the top-level folders and the main config files, so you know what work happens here.
2. List every skill I have: the ones you can see in this session, plus every folder in ~/.claude/skills and .claude/skills. Leave out plugin skills, whose names contain a colon (like vercel:deploy). I manage those in /plugin.
3. Read skillOverrides in .claude/settings.local.json in this folder and in ~/.claude/settings.json, so you know each skill's current state.
4. Show me a table: skill, current state, suggested state (on, name-only, user-only or off), and a one-line reason. My rule: relevant and helpful to this project means on. Flag any skills that do the same job, like several writing skills, and suggest keeping one on.
5. Stop there and change nothing until I answer.
6. When I say yes, or give you changes, merge the states into skillOverrides in .claude/settings.local.json in this folder. Write user-only as "user-invocable-only". Keep every other key and entry in that file, and do not touch ~/.claude/settings.json.
7. Then tell me to run /skills to check the result.
```

**Your part:** read the table and change any row you disagree with. Then say yes, approve the file edit when Claude Code asks, and open /skills to check the result. Read every row before you agree.

STEP 4

# Check and adjust in /skills

Open the panel. Each row shows a skill, its state, where it came from, and an estimate like **~340 tok**: what that skill's listing entry costs.

Type into Claude Code

```
/skills
```

| Key | What it does |
|---|---|
| **Type** | Filters the list by name. |
| **t** | Sorts by token count, so the costly skills rise to the top. |
| **Space or Enter** | Cycles the selected skill to its next state. |
| **Esc** | Saves and closes, writing skillOverrides to .claude/settings.local.json in this folder. |

### How Jason decides

"Is it relevant and helpful to this project? If yes, on. If no, off."

Name-only and user-only cover the skills in between: a skill whose name says enough, or one you want only when you type it.

**Your part:** decide each one against the four states table at the front of this guide, then press Esc to save.

STEP 5

# Break up stacks

Keep one skill per job on in a project. When two skills promise the same outcome, both descriptions compete for Claude's attention on every turn, and Claude may load the wrong skill or miss one that would help.

### Warning: overlapping skills

Jason had **humanizer**, **copywriting** and **the-writer** all on in one project. They stacked and collided, and in his experience the writing came out worse.

Pick the one that fits this project, leave it on, and set the others to user-only so you can still call them by name.

**Your part:** scan your list for skills that do the same job, like writing, reviewing or planning, and keep one of each on.

STEP 6

# Turn off plugins this project does not use

Plugin skills show a lock in /skills and cannot be cycled there, so you switch the whole plugin. In /plugin, open the **Installed** tab. A **Not used recently** group points at likely candidates. Press Enter on a plugin to enable, disable or uninstall it.

**Type into Claude Code**

```
/plugin
```

To turn a plugin off for this project only, run this from the project folder. It writes to the same .claude/settings.local.json.

**Run in a terminal, from the project folder**

```
claude plugin disable <plugin>@<marketplace> --scope local
```

**Your part:** swap in the plugin and marketplace names as /plugin shows them. Leave on anything this project uses. Disable keeps a plugin installed, so you can turn it back on later. Uninstall removes it.

STEP 7

# Set a lean default for everywhere else

/skills saves to the project folder, so the global default goes into **~/.claude/settings.json** by hand. On Windows that file is C:\Users\<you>\.claude\settings.json.

Add a skillOverrides block and merge it into the file you already have. Keep every other key. Swap in your own skill names and states.

**Add by hand to ~/.claude/settings.json**

```
{
  "skillOverrides": {
    "some-skill": "off",
    "another-skill": "name-only",
    "a-third-skill": "user-invocable-only"
  }
}
```

## How the two files work together

- **Both apply, skill by skill.** The global file sets the default; each project's .claude/settings.local.json adds its own choices on top.
- **The project wins a tie.** When both files set the same skill, the project's local file takes precedence, so a project can turn back on a skill that is off everywhere else.
- **Yours, or the team's.** .claude/settings.local.json is yours only. A project's .claude/settings.json is shared with the team; this guide never writes to it.

**Your part:** keep the file valid JSON, then open /skills in a project and confirm each skill shows the state you expect.

STEP 8, OPTIONAL

# The other two fixes from the reel

**ponytail** pushes Claude toward the simplest solution that works, the fix for reinventing the wheel on every small change. **humanizer** strips the patterns that make text read as machine-written.

## ponytail

It installs as a plugin, so it shows a lock in /skills. Source: github.com/DietrichGebert/ponytail

**Type into Claude Code, as two separate messages**

```
/plugin marketplace add DietrichGebert/ponytail
```

```
/plugin install ponytail@ponytail
```

### humanizer

Source: github.com/blader/humanizer. Leave off --global to install it for the current project only. The npx command needs Node.js.

**Run in a terminal**

```
npx skills add blader/humanizer --global
```

**Or install it as a plugin: type into Claude Code, as two separate messages**

```
/plugin marketplace add blader/humanizer
```

```
/plugin install humanizer@humanizer
```

**Your part:** install one or both, then give each one the state it earns in this project.

# When something goes wrong

| What you see | What it means | The fix |
|---|---|---|
| **A skill that fits the job never fires** | The listing may have overflowed and dropped its description, or another skill overlaps it. | Set idle skills to name-only or off, and keep one skill per job on. Start Claude Code with claude --debug to see whether descriptions are being dropped. |
| **Your /skills changes show up in the wrong project** | Claude Code started in a different folder, so the file landed there. | Close Claude Code, move into the project folder, start it again, and redo the changes in /skills. |
| **A skill shows a lock and will not cycle** | Most often it came with a plugin, and /skills cannot cycle plugin skills. | Turn the plugin off in /plugin, or for this project only with claude plugin disable <plugin>@<marketplace> --scope local. |
| **You need a skill here that is off globally** | The global default in ~/.claude/settings.json turned it off. | Set it to on in /skills inside this project. The project's local file takes precedence. Confirm the state in /skills. |
| **Edits to ~/.claude/settings.json do not show up** | The file may no longer be valid JSON, often from a missing comma or brace. | Ask Claude Code to check that the file is valid JSON and fix only the syntax, keeping every key. Then restart Claude Code and check /skills. |
| **/skill-doctor is an unknown command** | Your Claude Code is older than v2.1.252. | Run claude update in a terminal. Until then, /doctor estimates the listing cost. |
| **Writing gets worse with several writing skills on** | The skills stack and pull in different directions. | Keep one writing skill on and set the others to user-only. |

To undo everything: open /skills and cycle each skill back to on, or delete the skillOverrides entries from .claude/settings.local.json and ~/.claude/settings.json. Re-enable plugins in /plugin.

KEEP GOING

## The guide page

Every step, prompt and command in this PDF lives on the page too, with the latest checked facts: graniteai.co/tools/clean-up-claude-code-skills

We manage our own Claude Code skills this way. See what else GraniteAI makes: graniteai.co/tools/clean-up-claude-code-skills#products