

Ponytail: make Claude Code write less code

The plugin from the Reel. Ponytail makes Claude check what already exists before it writes anything, so you get smaller diffs and a smaller bill. Here's the 2-line install, the commands, what the numbers really say, and when it helps most.

Before you start: You need Claude Code (paid Claude plan or Console account) and Node.js on your PATH. Ponytail itself is free and open source (MIT).

Setup

- 1 In Claude Code, add the marketplace (send this as its own prompt)

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

- 2 Then install the plugin (a second, separate prompt)

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

- 3 Start a new session and check it's on

```
/ponytail
```

All 4, step by step

[#1](#) What Ponytail actually does

[#2](#) Install it in 2 prompts

[#3](#) The commands worth knowing

[#4](#) What the numbers really say

#1

What Ponytail actually does

Ponytail is a ruleset that the plugin injects into Claude Code on every turn. Before Claude writes code, it has to climb a ladder and stop at the first rung that works: skip it, reuse what's in the codebase, use the standard library, use a native platform feature, use a dependency you already have, write one line. Only then does it write new code. Ask for a date picker and, instead of a 400-line component, you get the browser's own `<input type="date">`.

YOU NEED

- Claude Code
- Node.js on your PATH (the plugin runs two tiny hooks)

1. Does this need to exist? If not, skip it.
2. Is it already in this codebase? Reuse it.
3. Does the standard library do it? Use it.
4. Is it a native platform feature (like `<input type="date">`)? Use it.
5. Is it in a dependency you already installed? Use it.
6. Can it be one line? Write one line.
7. Only then: write the minimum that works.

Tip: It never cuts validation, error handling, security or accessibility. It's lazy about the solution, not about reading your code.

[Official docs →](#)

#2

Install it in 2 prompts

The plugin installs from inside Claude Code. It has to be two separate prompts, one after the other.

YOU NEED • Claude Code (terminal or the Desktop app's Code tab)

1. Open Claude Code in any project.
2. Send: `/plugin marketplace add DietrichGebert/ponytail`
3. Send, as a new prompt: `/plugin install ponytail@ponytail`
4. Start a new session.
5. Desktop app: you can also click + next to the prompt box → Plugins → Add plugin.

COMMAND

```
/plugin marketplace add DietrichGebert/ponytail  
/plugin install ponytail@ponytail
```

Tip: If every hook call shows "node: command not found", install Node.js. The rules still work without it, but the error goes away once Node is on your PATH.

[Official docs](#) →

#3

The commands worth knowing

Ponytail adds a few slash commands. Use the level switch to make it stricter or turn it off for a task where you want Claude to build big.

YOU NEED • Ponytail installed

1. `/ponytail lite`: builds what you asked, but suggests smaller alternatives.
2. `/ponytail full`: the default, enforces the whole ladder.
3. `/ponytail ultra`: aggressive, cuts speculative features.
4. `/ponytail off`: turns it off.
5. `/ponytail-review`: reviews your current diff for over-engineering and gives you a delete-list.
6. `/ponytail-audit`: does the same for the whole repo.

COMMAND

```
/ponytail-review
```

PROMPT

Review what you just wrote. Is there anything here the browser, the standard library or an existing file already does? Give me a delete-list.

Tip: If a command isn't found, type `/ponytail` and look at the suggestions. Some versions show plugin commands with a prefix, like `/ponytail:ponytail`.

[Official docs](#) →

#4

What the numbers really say

The author's benchmark (12 feature tasks on a real FastAPI + React repo, Claude Haiku 4.5) measured about 54% less code, 22% fewer tokens and 20% lower cost on average. The big wins come from over-build traps: a date picker went from 404 lines to 23, a color picker from 287 to 23. An independent test by JetBrains (80 paired tasks) found smaller but real savings: about 15% less code and 10% lower cost, with no drop in quality.

1. Expect big cuts on UI widgets and helper code.
2. Expect little change on code that's already lean.
3. Install it as a plugin. JetBrains found that just copying the skill file in, without the plugin's hooks, means it basically never activates.

Tip: Treat it as a habit-shaper for Claude, not a magic token switch.

[Official docs →](#)

Tip: Ponytail shines where Claude tends to over-build: UI widgets, helpers, wrappers, "utility" files. On code that's already minimal it changes almost nothing.