

WHY

CLAUDE IGNORES YOUR RULES

10 PLAYS TO MAKE THEM HOLD



Stop repeating yourself.
Make your instructions hold.

Stop repeating yourself to Claude.

"Claude ignores my rules" is the most common complaint about Claude Code. Most of the time the rule is in the wrong place: advice that should be a hook, a procedure that should be a skill, or a line lost in a 600-line file. These ten plays show you how CLAUDE.md, rules, skills and hooks work together, and which one to use for each instruction.

How to read it

Each play fits on one page: when to do it, the steps in order, the commands and files you use, the decision that stays yours, the usual mistakes and a simple way to begin. Each page names the Claude Code feature it relies on.

How to use it

Start with the three diagnose plays: they take an hour and tell you which of the other seven you need. Page 16 helps you pick, in order.

CONTENTS

–	Why Claude ignores your rules	OVERVIEW	03
–	Ten plays at a glance	OVERVIEW	04
–	Four layers, one system	OVERVIEW	05
01	Check what Claude actually loaded	DIAGNOSE	06
02	Audit your instruction files	DIAGNOSE	07
03	Find the rule Claude keeps breaking	DIAGNOSE	08
04	A CLAUDE.md under 200 lines	STRUCTURE	09
05	Path-scoped rules in .claude/rules	STRUCTURE	10
06	Turn procedures into skills that trigger	STRUCTURE	11
07	Hooks for the rules that must hold	ENFORCE	12
08	Hooks that check after every edit	ENFORCE	13
09	Keep rules alive in long sessions	ENFORCE	14
10	Share the setup with your team	ENFORCE	15
–	Where to start: your plan	CHECKLIST	16

Why Claude ignores your rules

Seven causes come up again and again in the official docs and in developer forums. Each play in this guide fixes one or more of them.

1

It is advice, not code

CLAUDE.md reaches Claude as a user message after the system prompt. Claude reads it and tries to follow it, but nothing forces it to.

2

The file is too long

Past about 200 lines, adherence drops. Important rules get lost among style notes and history.

3

Rules contradict each other

Your user file, the project file and a subfolder file can say different things. Claude picks one, and not always yours.

4

The rule is vague

"Write clean code" cannot be checked. "Run npm test before every commit" can.

5

The session got long

The more the context fills, the less weight an early instruction carries.

6

Compaction summarised it away

After /compact the root CLAUDE.md is read again, but rules you typed in chat and nested files may be gone.

7

The skill never triggered

Claude chooses a skill from its description alone. A vague description, or too many skills for the listing budget, and it is never loaded.

200

Lines, at most

The docs advise keeping each CLAUDE.md under 200 lines. Shorter files are followed more reliably.

100%

Only hooks guarantee

CLAUDE.md and skills guide Claude. Hooks are code that runs every time, whatever the model decides.

Sources: [code.claude.com docs \(memory, skills, hooks\)](https://code.claude.com/docs/memory) · [platform.claude.com skill best practices](https://platform.claude.com/skill/best-practices).
Checked on 22 October 2026.

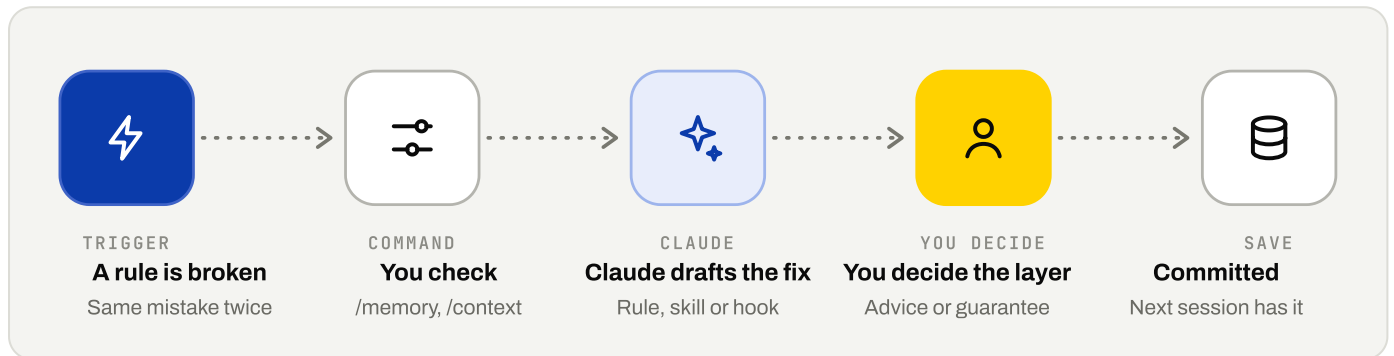
Ten plays for rules that **hold**

First find out what Claude really reads, then put each instruction in the right layer, then make the rules that matter impossible to skip.



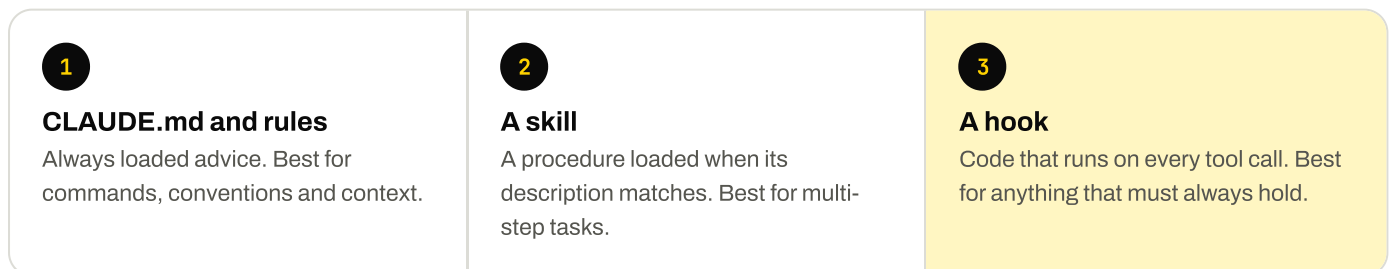
Four layers, one system

CLAUDE.md says how the project works. Rules add detail for one folder. Skills hold procedures Claude loads on demand. Hooks enforce what must never fail. Every play uses the same picture.



- 
Trigger
 What tells you to act: a rule ignored, a repeated correction, a new repo.
- 
Claude
 Claude reads, drafts or fixes. Useful, but never guaranteed.
- 
Save
 Saved to the repo or a notes file, so the next session has it.
- 
Wait
 Pauses until a date or a condition, such as a quarterly review.
- 
Command
 A command, file or script that runs the same way every time.
- 
You decide
 The point where you decide what the rule is and where it lives.
- 
Send
 Shared with the team or another repo.

THREE LEVELS OF CERTAINTY



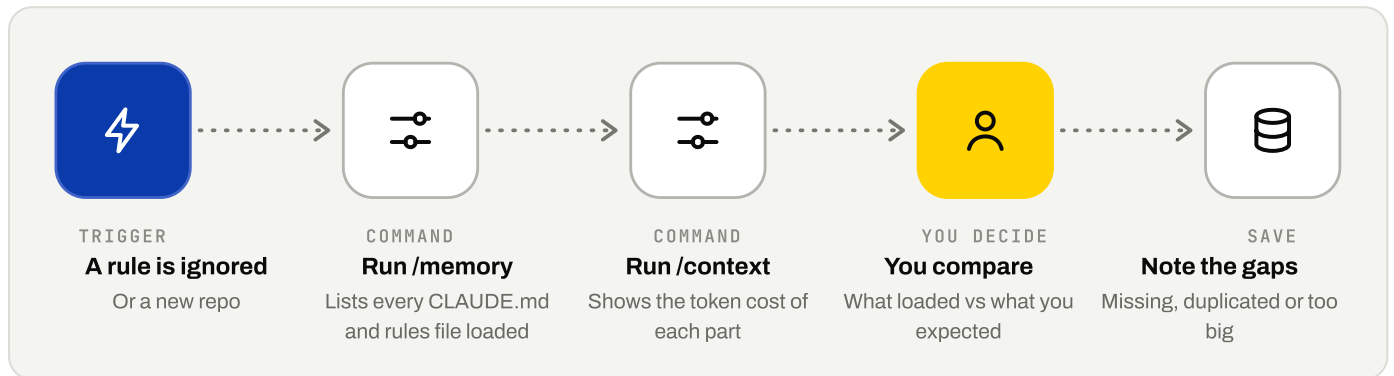


The golden rule
 If breaking a rule once is acceptable, write it down. If it is not, make it a hook.

01

Check what Claude actually loaded

Before you rewrite anything, find out which instruction files Claude has in context right now, and how much room they take.



WHEN TO DO IT

Claude ignores a rule, or you start in a new repository.

WHAT YOU USE

/memory

/context

InstructionsLoaded hook

**Your call**

You decide which files belong in the session.
Nothing changes until you edit a file.

WATCH OUT FOR

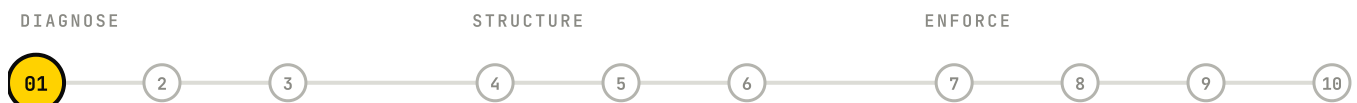
- ⚠ Assuming a subfolder CLAUDE.md is loaded at start: it loads only when Claude reads files in that folder.
- ⚠ Looking for AGENTS.md rules: Claude Code reads AGENTS.md only when no CLAUDE.md exists. Import it with @AGENTS.md.
- ⚠ Forgetting CLAUDE.local.md and ~/.claude/CLAUDE.md, which also load and can override you.

START SIMPLE

Open a fresh session in the repo, run /memory, and write down every file it lists. That list is your starting point.

WHERE IT SITS IN THE GUIDE

Feeds into 02 · Audit your instruction files

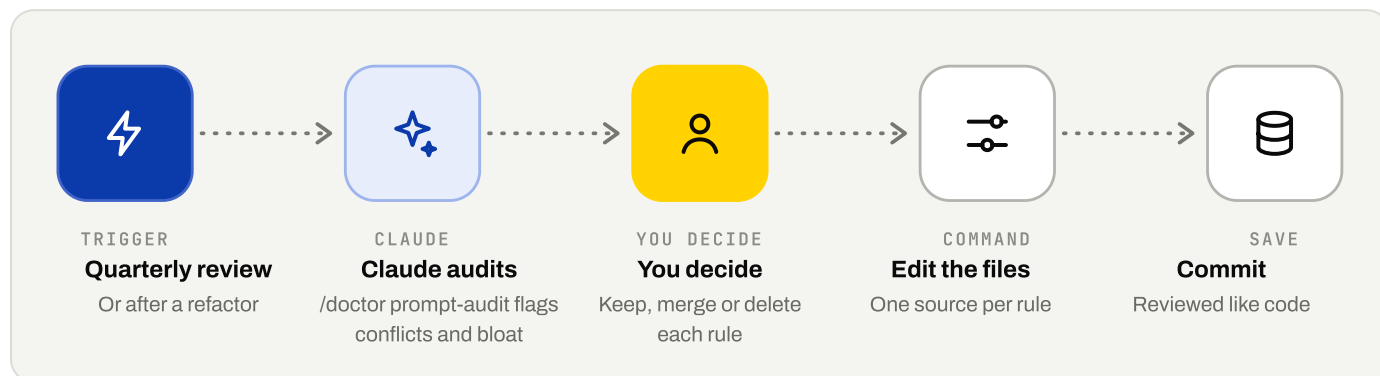


02

■ PLAY 02 · DIAGNOSE **/doctor prompt-audit**

Audit your instruction files

Read every instruction file as Claude does, then remove conflicts, duplicates and rules that no longer apply.



WHEN TO DO IT

Once per repository, then each quarter or after a big change.

WHAT YOU USE

`/doctor prompt-audit`

`git diff`

Your PR review

WATCH OUT FOR

- ⚠ The same rule in three files with three wordings: keep one, delete the others.
- ⚠ Rules about tools you stopped using months ago.
- ⚠ Comments for humans inside CLAUDE.md: HTML comments are stripped, so they cost nothing but help nobody either.



Your call

The audit only suggests. You choose which rule wins when two files disagree.

START SIMPLE

Paste all your instruction files into one conversation and ask Claude to list every pair of rules that contradict each other.

WHERE IT SITS IN THE GUIDE

Feeds into 03 · Find the rule Claude keeps breaking

DIAGNOSE

STRUCTURE

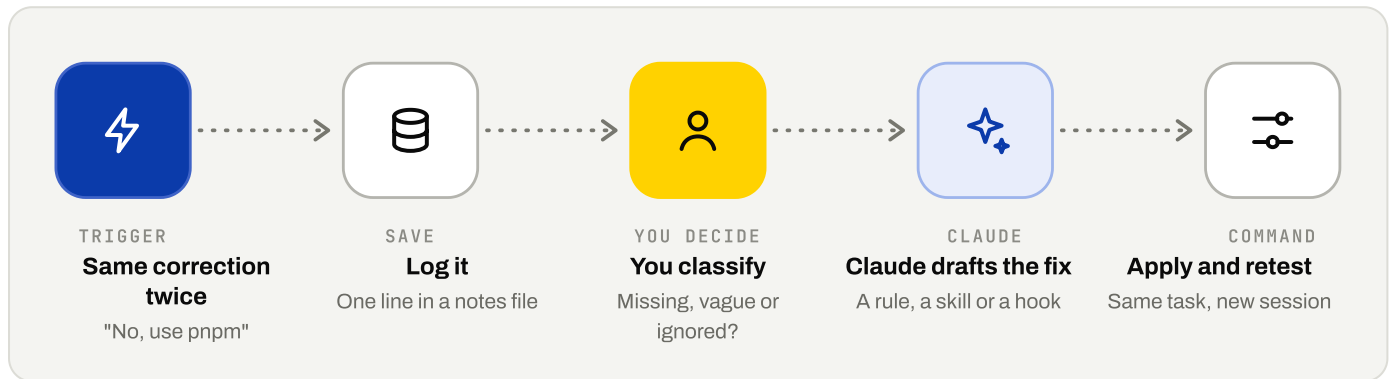
ENFORCE



03

Find the rule Claude keeps breaking

Stop guessing. Track the corrections you type by hand, and fix the instruction behind the ones that come back.



WHEN TO DO IT

You correct Claude for the same thing a second time.

WHAT YOU USE

A corrections.md file

shortcut to add a memory

/memory



Your call

You decide the right layer: a missing rule goes in CLAUDE.md, a procedure in a skill, a must-hold rule in a hook.

WATCH OUT FOR

- ⚠ Adding a line in capitals for every correction: the file grows and nothing improves.
- ⚠ Fixing it only in the chat: it is lost at the next session or compaction.
- ⚠ Not retesting in a fresh session, where the old context cannot help.

START SIMPLE

Keep a five-line log this week. By Friday you will know which three rules matter most.

WHERE IT SITS IN THE GUIDE

Feeds into 04 · A CLAUDE.md under 200 lines

DIAGNOSE

STRUCTURE

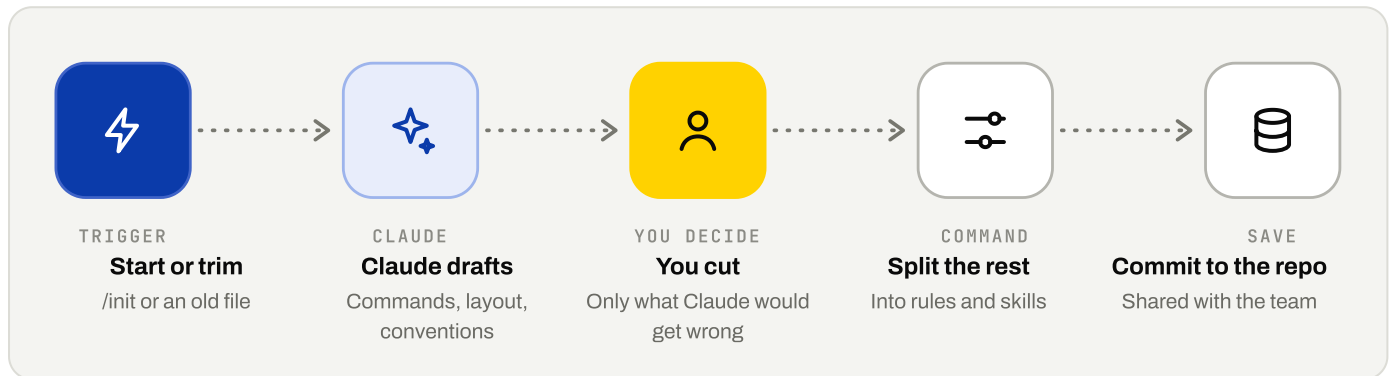
ENFORCE



04

A CLAUDE.md under 200 lines

One short project file with concrete, checkable rules: build and test commands, conventions and the things Claude cannot guess.



WHEN TO DO IT

You run /init, or your file has grown past 200 lines.

WHAT YOU USE

/init

./CLAUDE.md or ../.claude/CLAUDE.md

CLAUDE.local.md for personal notes

WATCH OUT FOR

- ⚠ Vague rules such as "be careful": replace them with a command or a file path.
- ⚠ Using @imports to save space: imported files still load in full.
- ⚠ Long explanations of why: one line of rule beats a paragraph of history.



Your call

Every line is yours to approve. If a rule cannot be checked, rewrite it or delete it.

START SIMPLE

For each line ask: would Claude make a mistake without it? If not, delete it.

WHERE IT SITS IN THE GUIDE

Feeds into 05 · Path-scoped rules in .claude/rules

DIAGNOSE

STRUCTURE

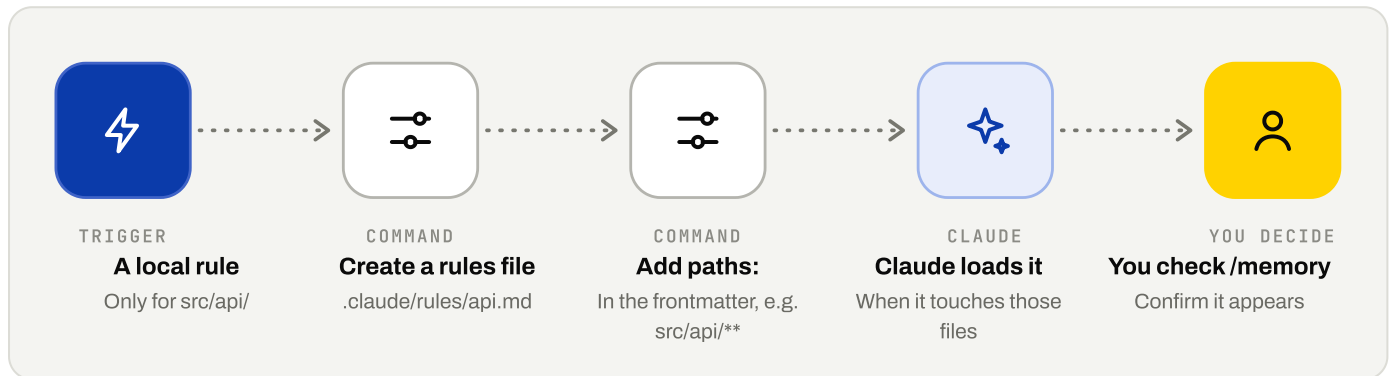
ENFORCE



05

Path-scoped rules in `.claude/rules`

Rules for one part of the code load only when Claude works there, so the main file stays short and the rule shows up when it matters.



WHEN TO DO IT

A rule applies to one folder or file type only, such as API handlers or tests.

WHAT YOU USE

`.claude/rules/``paths: frontmatter``/memory`

WATCH OUT FOR

⚠ Writing globs: as in other tools: Claude Code ignores it, use paths.

⚠ Too many tiny rules files that all load anyway because the paths are too wide.

⚠ Moving a rule here that must apply everywhere, such as how to run tests.



Your call

You set the paths. A rules file without paths: loads in every session, like `CLAUDE.md`.

START SIMPLE

Move one block from `CLAUDE.md` that only concerns tests into `.claude/rules/tests.md` with paths for your test folder.

WHERE IT SITS IN THE GUIDE

Feeds into 06 · Turn procedures into skills that trigger

DIAGNOSE

STRUCTURE

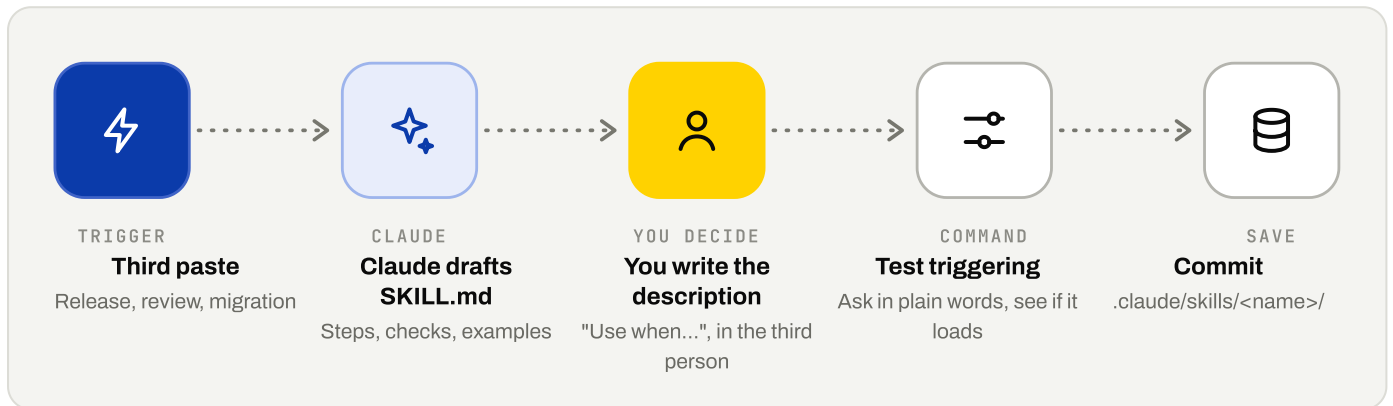
ENFORCE



06

Turn procedures into skills that trigger

Package a repeatable procedure as a skill whose description says exactly when to use it, so Claude loads it at the right moment.



WHEN TO DO IT

You paste the same procedure into the chat for the third time.

WHAT YOU USE

`.claude/skills/` `/skill-doctor`
`disable-model-invocation`



Your call

You decide which skills Claude may start on its own. Set `disable-model-invocation` for skills you want to run only by hand.

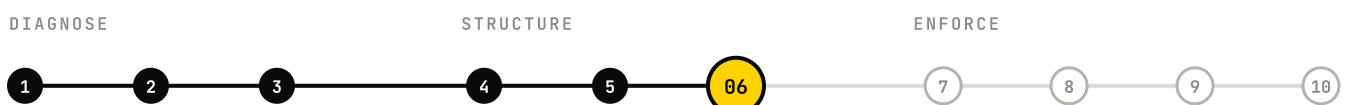
WATCH OUT FOR

- ⚠ A description that says what the skill is, not when to use it: it never triggers.
- ⚠ SKILL.md over 500 lines, or reference files nested two levels deep.
- ⚠ Expecting `allowed-tools` to restrict a skill: it only pre-approves tools.

START SIMPLE

Start the description with "Use when" and add the words your team actually types, then run `/skill-doctor`.

WHERE IT SITS IN THE GUIDE

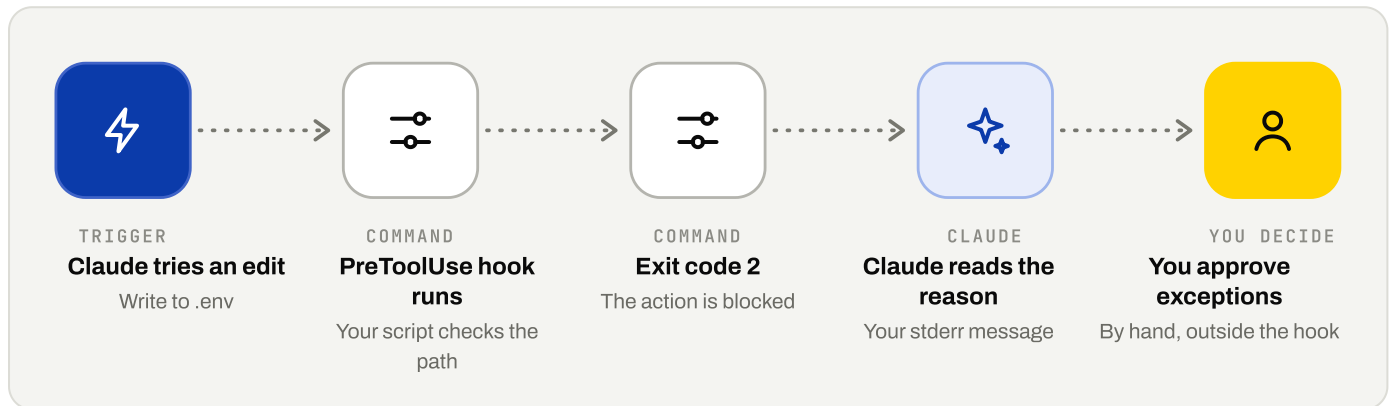


Feeds into 07 · Hooks for the rules that must hold

07

Hooks for the rules that must hold

Turn "never" rules into code. A hook runs before the action and can stop it, every time, whatever the context says.



WHEN TO DO IT

A rule where one miss is too costly: secrets, production, protected files.

WHAT YOU USE

`.claude/settings.json` hooks

matcher: Edit|Write

A small shell script

WATCH OUT FOR

- ⚠ A hook that blocks without a clear message: Claude retries the same thing.
- ⚠ A slow hook on every tool call: keep the matcher narrow.
- ⚠ Untested scripts: run them by hand with sample input before you rely on them.



Your call

The hook decides, not the model. Only you can change the script or the settings file.

START SIMPLE

Write one hook that blocks edits to .env and any file under secrets/, with a one-line reason.

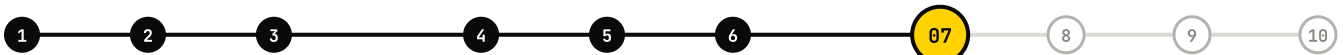
WHERE IT SITS IN THE GUIDE

Feeds into 08 · Hooks that check after every edit

DIAGNOSE

STRUCTURE

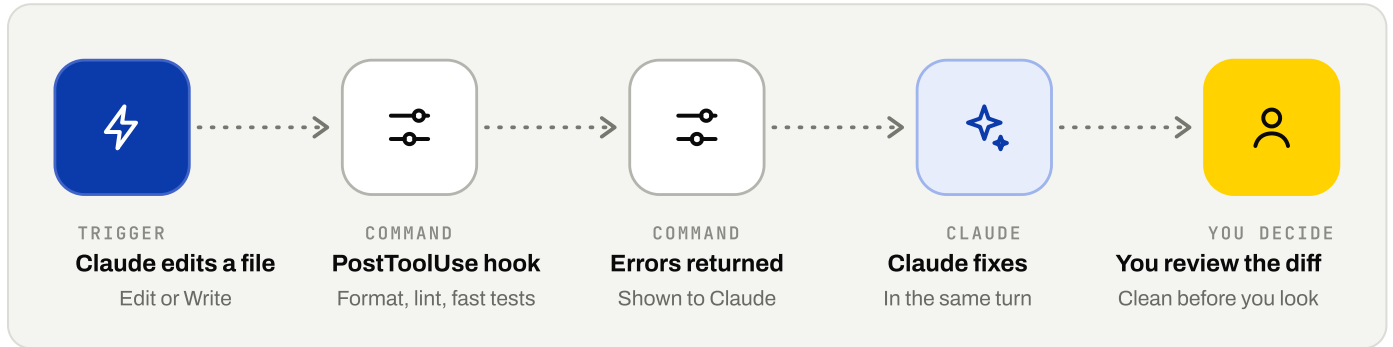
ENFORCE



08

Hooks that check after every edit

Run the formatter, linter or fast tests after each edit, and send the errors straight back to Claude to fix.



WHEN TO DO IT

Claude forgets to lint or format, or breaks a test it did not run.

WHAT YOU USE

Prettier or Black

ESLint or Ruff

Your fast test command

WATCH OUT FOR

- ⚠ Running the whole test suite after every edit: sessions crawl.
- ⚠ Silent formatters that change files Claude then edits again from memory.
- ⚠ Duplicating the rule in CLAUDE.md as well: the hook is enough.



Your call

You choose which checks run and how strict they are. The full test suite stays in CI.

START SIMPLE

Start with the formatter only. Add the linter once the formatter runs cleanly for a week.

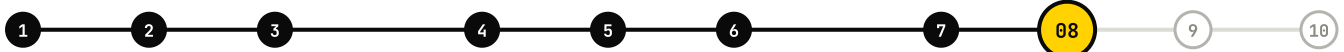
WHERE IT SITS IN THE GUIDE

Feeds into 09 · Keep rules alive in long sessions

DIAGNOSE

STRUCTURE

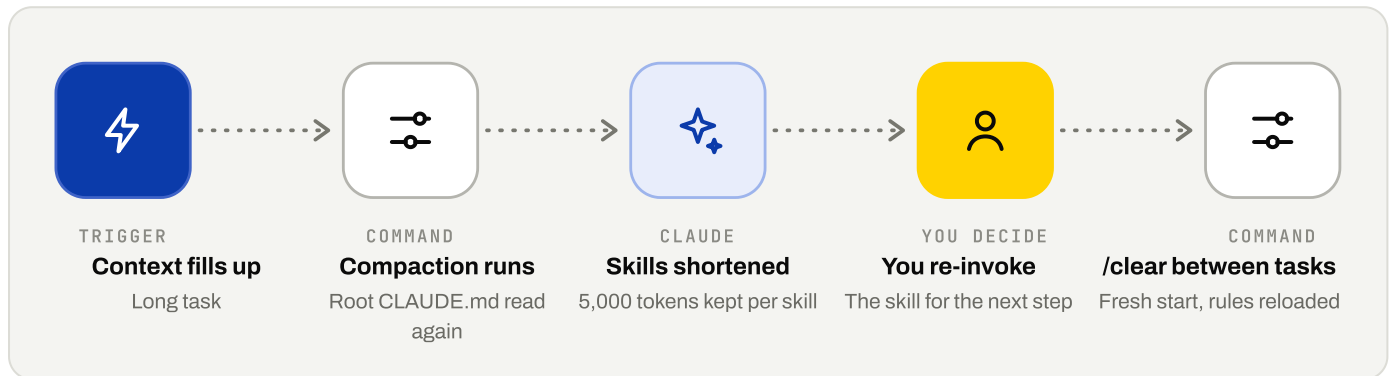
ENFORCE



09

Keep rules alive in long sessions

Rules fade as the context fills. Plan for compaction so the important ones come back on their own.



WHEN TO DO IT

A long task, a session that has run for hours, or a compaction.

WHAT YOU USE

`/compact``/clear``/context`

WATCH OUT FOR

- ⚠ Rules typed only in the chat: they may not survive compaction.
- ⚠ Key rules buried in a nested file that is not reloaded after `/compact`.
- ⚠ One endless session for a whole day of different tasks.



Your call

You decide when to clear and when to compact. One task per session is the simplest rule.

START SIMPLE

Put your five most important rules at the top of the root CLAUDE.md, and run `/clear` when you switch task.

WHERE IT SITS IN THE GUIDE

Feeds into 10 · Share the setup with your team

DIAGNOSE

STRUCTURE

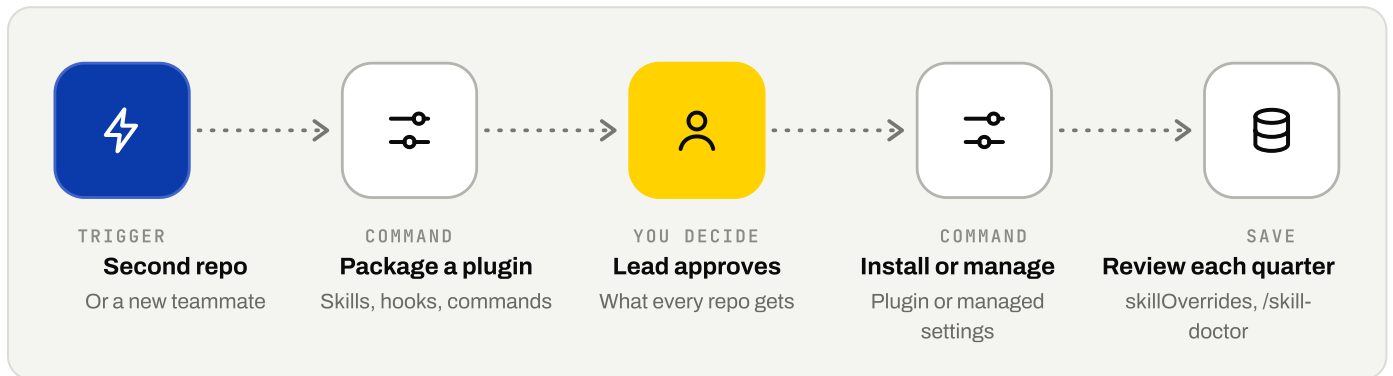
ENFORCE



10

Share the setup with your team

When a second repository or a new teammate needs the same setup, package your rules, skills and hooks once and install them everywhere.



WHEN TO DO IT

A second repo, a new hire, or the same skill copied by hand.

WHAT YOU USE

Plugins

Managed settings

skillOverrides

WATCH OUT FOR

⚠ Installing public skills without reading them: some carry hidden instructions or scripts.

⚠ Too many shared skills: the listing budget drops the least used descriptions.

⚠ Personal preferences in the shared plugin: keep them in ~/.claude/.



Your call

One owner reviews every shared skill and hook before it ships. Treat them like code.

START SIMPLE

Move your three most used skills and your .env hook into one internal plugin, and install it in a second repo.

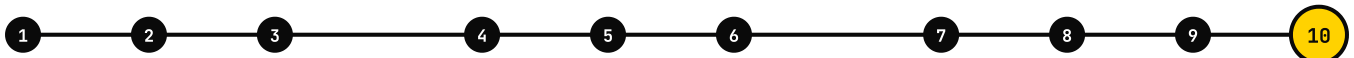
WHERE IT SITS IN THE GUIDE

Closes the loop: review the whole setup every quarter (play 02)

DIAGNOSE

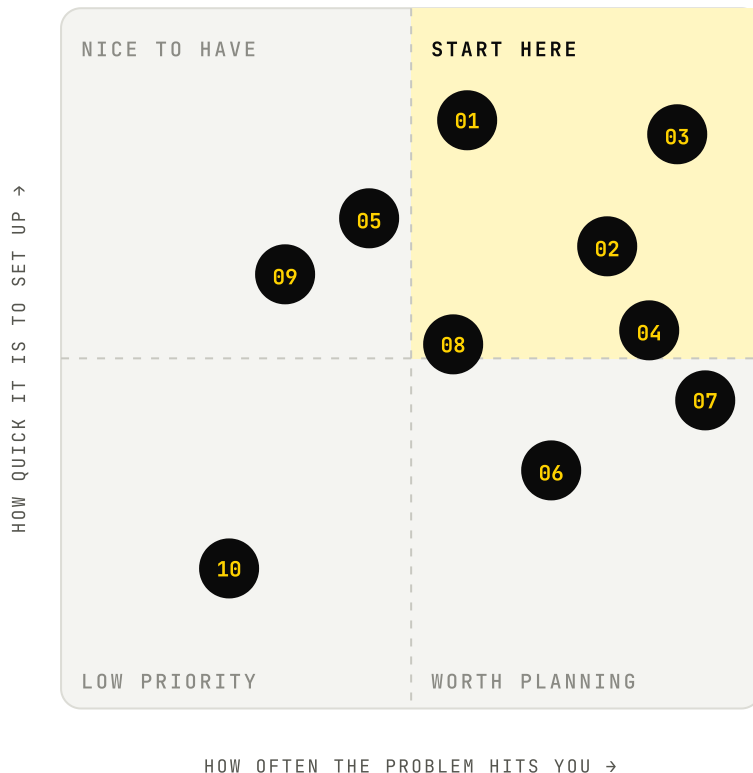
STRUCTURE

ENFORCE



Where to start: your plan

Start with what hits you most often and is quickest to set up. The diagnose plays come first because they tell you which others you need.



- 01 Check what Claude actually loaded
- 02 Audit your instruction files
- 03 Find the rule Claude keeps breaking
- 04 A CLAUDE.md under 200 lines
- 05 Path-scoped rules in .claude/rules
- 06 Turn procedures into skills that trigger
- 07 Hooks for the rules that must hold
- 08 Hooks that check after every edit
- 09 Keep rules alive in long sessions
- 10 Share the setup with your team

Suggested placement for a typical small team. Move the dots to match your own repositories.

YOUR SIX-STEP PLAN

- ☐ Open a fresh session, run /memory and /context, and list every file that loads (play 01).
- ☐ Log every correction you type for one week (play 03).
- ☐ Turn the procedure you paste most into a skill, and add one hook for your hardest rule (plays 06 and 07).
- ☐ Run /doctor prompt-audit and remove conflicts and duplicates (play 02).
- ☐ Cut CLAUDE.md under 200 lines and move folder rules to .claude/rules (plays 04 and 05).
- ☐ Retest your logged corrections in a new session. Each one that still happens needs a hook.

■ WHAT'S NEXT

Rules that hold. Code you can trust.

■ LEARN IT

AI training

Hands-on Claude Code sessions in Seville and Málaga, built around your own repositories.

■ BUILD IT WITH US

Automation services

We set up CLAUDE.md, skills, hooks and plugins for your team, and review them with you.

■ EVERY MONDAY

The Monday playbook

One practical AI workflow in your inbox each week. Free, unsubscribe in one click.



Claude and Claude Code are names of Anthropic. Generative Z is independent of Anthropic. Commands and settings change between versions: check code.claude.com/docs before you rely on them. Read every public skill, hook and plugin before you install it, and keep secrets out of instruction files.