EcomLabs

STEP-BY-STEP GUIDE

A short CLAUDE.md: procedures to skills, rules to hooks

A short CLAUDE.md template, a deploy skill that runs only when you invoke it, a hook that enforces a rule and a Node script that flags which lines to move out of the CLAUDE.md you already have. With a checklist.

FOR	Projects that use Claude Code and have a CLAUDE.md that grew too long
TESTED WITH	Node 22.22.3 · hook and script tested with hand-made JSON and sample files · YAML checked with the yaml package
UPDATED	November 19, 2026
DOCS	code.claude.com/docs/en/memory · /skills · /hooks
ARTICLE	ecomlabs.dev/g/reglas-en
AUTHOR	Luis Morales · @luismoralesdev

Contents

Step 01	The CLAUDE.md template
Step 02	A procedure as a skill
Step 03	A must-hold rule as a hook
Step 04	Prune the CLAUDE.md you already have

STEP 01

The CLAUDE.md template

Save it as `CLAUDE.md` at the repo root and replace everything between `< >`. It loads in full every session, so every line has to be useful to all of them. Target: under 200 lines, counting `@` imports, which load together with the file. Leave out the directory tree, the dependency list and anything Claude reads from the code.

```
# <Project name>

<One sentence: what it is and what it is built with.> Package manager: <pnpm>.

## Commands
- Dev: <pnpm dev>
- Tests: <pnpm test> · Types: <pnpm typecheck>
- Before you call a task done: <pnpm test and pnpm typecheck>

## Conventions
- <A decision you cannot see in a single file, written so it can be checked.>
- <Example: prices are stored as integer cents and formatted only in the UI.>

## Known traps
- <Something that breaks and the code does not show. Example: the seed wipes the orders table.>

## Where things live
- Procedures: .claude/skills/ (migrations, release).
- Deploys: I run /deploy myself; do not deploy.
- Folder rules: .claude/rules/.

<!-- For people (Claude never sees this): must-hold rules are hooks
     in .claude/settings.json; personal notes go in CLAUDE.local.md. -->
```

STEP 02

A procedure as a skill

Save it as `.claude/skills/deploy/SKILL.md`. With `disable-model-invocation: true`, Claude cannot invoke it on its own and its description takes no context: it runs only when you type `/deploy`. For procedures without side effects, such as adding a migration, drop that line. Replace the commands with yours:

```
---
name: deploy
description: Deploys the current commit to production after the checks pass.
disable-model-invocation: true
---

# Deploy
```

```
1. Run git status. If there are uncommitted changes, stop and tell me.
2. Run pnpm test and pnpm build. If anything fails, stop and show the error.
3. Run bash scripts/deploy.sh and wait for it to finish.
4. Check that the production URL answers with 200.
5. Reply with the deployed commit and the result of each step.
```

In a skill Claude can invoke, the `description` is the only part it sees every session: put what it does first and when to use it second. After compaction Claude Code re-attaches the first 5,000 tokens of the body, so what matters goes at the top.

STEP 03

A must-hold rule as a hook

"Never use npm" in CLAUDE.md is a request. This hook enforces it. Register it in `.claude/settings.json`:

```
{
  "hooks": {
    "PreToolUse": [
      {
        "matcher": "Bash",
        "hooks": [
          {
            "type": "command",
            "command": "node",
            "args": ["${CLAUDE_PROJECT_DIR}/.claude/hooks/no-npm.mjs"]
          }
        ]
      }
    ]
  }
}
```

The `command` + `args` form uses no shell: Claude Code substitutes `${CLAUDE_PROJECT_DIR}` as plain text, so it works the same under Git Bash or PowerShell. The script, in `.claude/hooks/no-npm.mjs`:

```
// PreToolUse hook: this repo uses pnpm, so block any Bash command that runs npm or npx.
import { readFileSync } from "node:fs";

let input;
try {
  input = JSON.parse(readFileSync(0, "utf8")); // Claude Code sends the event as JSON on stdin
} catch {
  process.exit(0); // unreadable input: no decision, the normal permission flow continues
}

const command = input.tool_input?.command ?? "";
// npm or npx at the start of the command or after ; & | ( or a newline
if (/(^|[;&|(\n]\s*)(npm|npx)(\s|$)/.test(command)) {
```

```
  console.error("This repo uses pnpm. Use pnpm instead of npm, and pnpm dlx instead of npx.");
  process.exit(2); // exit 2 on PreToolUse: the call is blocked and Claude reads stderr
}
process.exit(0);
```

Test it before you rely on it, by piping JSON in by hand:

```
echo '{"tool_name":"Bash","tool_input":{"command":"cd app && npm test"}}' | node .claude/hooks/no-npm.mjs;
echo "exit $?"
# expected: the pnpm message and exit 2
echo '{"tool_name":"Bash","tool_input":{"command":"pnpm test"}}' | node .claude/hooks/no-npm.mjs; echo "exit
$?"
# expected: exit 0
```

On exit 2 the call is blocked and Claude reads stderr; any other non-zero code does not block. It does not catch `npm` inside another script, or `sudo npm`: it is a guard, not a sandbox.

STEP 04

Prune the CLAUDE.md you already have

Go through your CLAUDE.md line by line (or run the article's `claude-md-check.mjs` script, which flags candidates) and pick a destination for each one:

IF THE LINE IS…	IT GOES TO
A command, convention or fact every session needs	CLAUDE.md
A procedure or reference material	`.claude/skills/`
A procedure with side effects	Skill with `disable-model-invocation: true`
A rule for one folder or file type	`.claude/rules/` with `paths`
A rule that must hold every time	Hook, or a `deny` rule in permissions
Something Claude can derive from the code	Deleted
A personal preference	`CLAUDE.local.md`

Then run `/doctor`, which proposes trims, and check `/context all` in a new session.