PathionAI

FORGE PATHS. DELIVER RESULTS.

COMPLETE BEGINNER'S GUIDE

Make your AI agent prove it finished

unlazy is a free add-on for Claude Code and Codex. It makes the agent write a checklist of testable outcomes before it starts, then run those tests before it is allowed to say the work is done.

WHAT IT IS

A set of instructions your agent reads, plus a small program that checks its work

AI coding agents have a habit: they stop early and report success. The code looks finished, the summary sounds confident, and three of the seven things you asked for were never done. unlazy fixes the incentive. Before touching any code, the agent writes a file called `GATES.md` listing each outcome and the exact command that proves it. A gate only counts as passed when that command runs and succeeds. "Done" stops being the agent's opinion and becomes something a script decides.

WHAT IT DOES

Four things, all of them boring on purpose

- **Forces a checklist up front.** Every outcome gets a command that can fail.
- **Refuses fake evidence.** A ticked box with no recorded output still counts as unfinished.
- **Splits big jobs into a tree.** Each branch gets its own checklist and its own owned files.
- **Re-runs the checks at the end**, rather than trusting results from earlier in the session.

WHAT IT REPLACES

The "are you sure you actually did all of it?" loop

Right now you catch half-finished work by reading the diff yourself and sending the agent back three or four times. That is ten to twenty minutes of your attention per task, and you only catch what you happen to look at. unlazy moves that check into a command the agent has to pass. It costs nothing, has no paid tier, and installs in about a minute.

BEFORE YOU START

You need Node version 16 or newer

Node is the program that runs unlazy's checker. Most people who already use Claude Code have it. Nothing else is required — unlazy installs no other packages.

→ **To open Terminal:** press ⌘ **+ Space**, type *Terminal*, press Enter. A plain black-or-white window with text in it will open. That is correct — it is meant to look like that.

→ **To check Node:** type this and press Enter.

```
node -v
```

If it prints something like **v20.11.0**, you are fine — anything 16 or higher works. If it says "command not found", go to **nodejs.org** and download the LTS version, run the installer, then close and reopen Terminal and try again.

STEP 1

Install unlazy

1 Paste this into Terminal and press Enter

```
npx skills add Leonxlnx/unlazy -g
```

The first time, it will ask permission to download a helper called *skills*. Say yes. It then detects which AI agents you have installed and asks which to add it to. The **-g** matters: it installs unlazy once for every project instead of only the folder you are standing in.

2 Check it landed

```
ls ~/.claude/skills/unlazy
```

You should see a list including **SKILL.md** and **scripts**. If you use Codex instead of Claude Code, the folder is **~/.codex/skills/unlazy** and every command below works the same with that path swapped in.

STEP 2

Write the checklist

In the project you are working on, create a file named **GATES.md**. Each entry is one outcome you could be wrong about, plus the command that would catch you. Ask your agent to write it and it will follow this shape:

```
- [ ] G1: pricing page renders all three tiers
  CHECK: node scripts/verify-pricing.mjs
  EXPECT: pricing verification passed
  EVIDENCE: pending
```

CHECK is the command. **EXPECT** is a phrase that only gets printed when the command genuinely succeeds. A gate passes only if the command exits cleanly *and* that phrase appears. Never copy a number you were given into EXPECT — measure it instead, or you are just proving the agent can repeat itself.

STEP 3

Read it before you run it

This is the step not to skip. Those **CHECK** lines are real shell commands that run with full access to your files, your environment and your network. If the agent wrote them, or you copied a checklist from someone else's project, read every line first.

1 Inspect without running anything

```
node ~/.claude/skills/unlazy/scripts/gate-check.mjs --status GATES.md
```

This is the only mode that never executes a command. Read what it prints.

2 Once you understand every command, approve and run

```
node ~/.claude/skills/unlazy/scripts/gate-check.mjs --approve GATES.md
```

Approval is remembered per exact command. Change the command and it asks again. Approval is permission, not a sandbox — it does not limit what the command can do.

STEP 4

Hand it to the agent

1 For a normal task, just name the skill

```
/unlazy add rate limiting to the API and verify every route
```

One agent, one checklist. Start here — this is most of what you will ever need.

2 For a big job, ask for a tree

```
/unlazy tree 4 refactor the payment module
```

It splits the work four layers deep and gives every piece its own checklist and its own owned files.

3 Only if you want several agents at once, add this to the same message

```
You may run sub-agents in parallel for this task. Use rolling
dispatch: start each leaf as soon as its dependencies are
VERIFIED, rather than waiting for a whole batch to finish.
Before dispatching any leaf, confirm it declares disjoint OWNS:
paths and that its --claim succeeded.
```

Claude Code will not spawn parallel agents unless you say so, which is why this exists. Do not use it on small tasks — you pay for every agent, each one starts with no memory of the others, and one agent working straight through usually does better work.

4 Before you accept the result, re-run the checks yourself

```
node ~/.claude/skills/unlazy/scripts/gate-check.mjs --reverify GATES.md
```

Old evidence in the file is not proof it still passes. That's it.

GOOD TO KNOW

Pin a version. There is no tagged release yet. If you want it to stop changing under you, install it, then note the commit you are on. **The author is honest about the evidence** — the project used to claim a measured improvement and removed it because the data was not reproducible. Treat it as a discipline, not a guaranteed score. **There is an optional hook** that blocks Claude Code from ending its turn while gates are unmet; install it only when you want that, with *scripts/install-hooks.mjs*. **To remove everything**, delete the folder *~/.claude/skills/unlazy* and the folder *~/.unlazy*.