My Claude Code Setup: Auto Mode by Default

Why My Guardrails Aren't Permission Prompts

Idaliya Grigoryeva

UC San Diego

July 2026

Agentic Tools Session · UCSD Economics

What You'll Leave With

- I'm the **non-expert** here: no terminal, no sandbox, no cluster, nothing at the OS level
- Everything runs in the **Claude desktop app, Code mode**, straight into my real project folders

By the end, you should be able to:

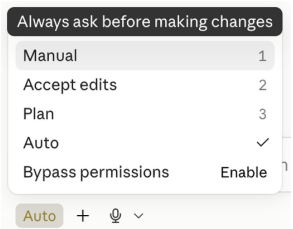
1. Tell the **four permission modes** apart, and pick one on purpose
2. Say what **Auto** does, and what it doesn't
3. Set up the **three config layers**, and know what goes in each
4. Decide whether a loose setup is safe **for your folder**

Four Permission Modes & My Preference

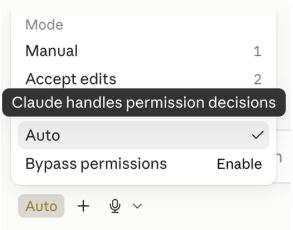
Mode	What it does	Me
Manual	Asks before every change	Never
Accept edits	Edits yes, commands ask	By accident, early on
Plan	Read-only, no changes	Never
Auto	Decides what's safe to run	**Always**
Bypass	Skips everything	Off, and stays off

- I ran **Accept edits** for a while without realizing it
- It was **extremely annoying**: it still stopped at every command, which is most of a real run

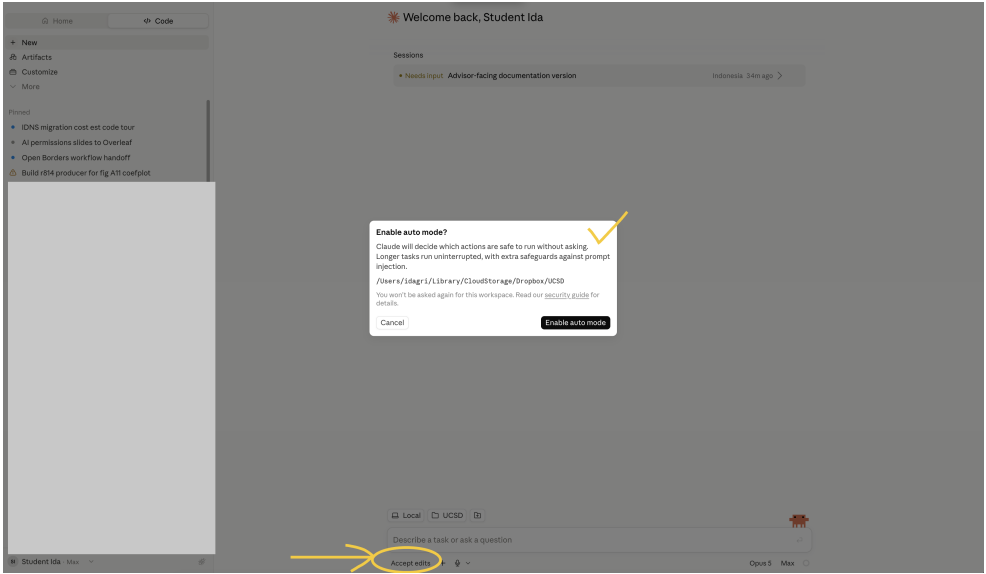
The options



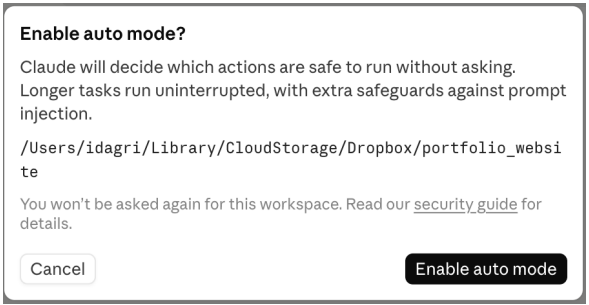
My choice



Where the Permission Controls Live



What Auto Actually Is



- It is **not** the same as bypassing permissions, which I leave off
- Claude still judges each action, and destructive or irreversible ones still surface
- It is set **per workspace**, so one folder at a time, not a global switch

Why I default to it

- Infeasible when multi-tasking: multiple projects, several tasks per project, running concurrently
- By the 3rd prompt, clicking allow without reading it, stalling progress against limited attention
- Better to put the effort into the `CLAUDE.md` **rules** than into the prompts

My Setup Preferences

What I run

- Claude **desktop app**, **Code mode**, no terminal
- **Max plan (x20)**, model use varies by task
- Straight into my **real project folders**, Dropbox-backed
- One project per folder, with its data and its code

What I deliberately don't have

- No sandbox
- No identified or private data in *any* folder Claude can see

The condition that makes this reasonable

- Manual review is worth its cost only when **I can't roll back**
- For how I work, that is almost never
- **Change that condition and my answer changes too**

The Rule That Does the Work: Never Delete

One rule in my global `CLAUDE.md` carries more weight than every permission prompt I turned off.

Rule (1), from my global `CLAUDE.md`

Do NOT delete anything that already exists when you work, only create new files. If you created something that is no longer relevant, **move it to a subfolder** `_archive_claude` that you can create in your working directory, **but do not delete**

- First written as **profile settings in the Claude web app**, back when I worked in chat
- In Claude Code they became a real file, `~/.claude/CLAUDE.md`, read on **every** project

Two more, and that's the whole net

- **Rollback**: git plus Dropbox version history, so a bad run gets reverted, not debugged
- **Written guidance**: a `CLAUDE.md` per project and a running lessons-learned file

I only run permissions this loose **because** these are in place first

Where the Settings Actually Live

.claude/settings.local.json, per project

```
"permissions": {
  "allow": [
    "Bash(python3:*)",
    "Bash(mkdir *)",
    "Bash(find *)",
    "Bash(curl *)",
    "Write(**)",
    "Edit(**)"
  ]
}
```

Three layers, in order

- `~/.claude/CLAUDE.md`
  my nine rules, every project
- `<project>/CLAUDE.md`
  conventions for that project only
- `<project>/.claude/settings.local.json`
  what is pre-approved, no prompt

This **fills itself in**: every "always allow" you click lands here

Most of my remaining prompts weren't a permissions problem, they were a **workflow** problem

Where This Would Break

- **With privacy-sensitive data in the folder.** Nothing here applies: no identified records sit anywhere Claude can see, and that is the precondition for every slide before this one
- **When the damage is permanent.** A deleted file is fixable, but leaked PII or a published API key is not, and rollback does nothing for either
- **With prompt injection.** **Read** access alone becomes the ability to act, which is why read permissions matter more than they look

What would make me tighten up

- PII in the project folder
- Credentials or API keys in the repo
- Anything that can **send or publish on my behalf**
- Any of those and I'd want a sandbox

Student data? Thinking with Agents: Agentic AI for Teaching covers de-identification in practice

Worth Knowing When Running on Auto

- **Traffic goes out, and it adds up.** OCR-ing images and downloading datasets pull real bandwidth, which on a **hotspot** is a data plan
- **Retries cost more.** A step that fails and retries spends its tokens again, so the credit ceiling can arrive mid-task
- `caffeinate` **keeps a long run alive** by stopping the Mac from idle-sleeping while the job is going
- **Closing the lid is a separate setting.** `caffeinate` covers idle sleep, not the lid, and Claude can set that one up for you
- **Make partial progress retrievable.** Have subagents **write output to disk as they go**, so a run that dies at 80% still leaves the 80%

Same instinct as the never-delete rule

- Assume the run breaks somewhere, and make the breakage **cheap**
- Write it into `CLAUDE.md` **before** you need it

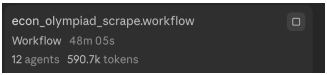
Token Use and Parallel Work

There is no live usage meter

- Claude **cannot** track its own token or traffic usage while it works, there is no running counter it can check mid-task
- But it **can estimate up front**: ask roughly how much this will take before kicking off a long job
- Cheap habit, and about the only budget signal you get

Background tasks vs. spin-off chats

- **Background tasks** run alongside what you are doing and report back into the same session
- **Spin-off chats** are a separate thread with their own context, for work that shouldn't share the current one



Still going at 48 minutes

What to Take Away

1. **Turn Auto on.** It lets a long run finish while you do something else, and Claude still judges every action and surfaces the destructive ones
2. **Put version control underneath it first.** Git, Dropbox history, or both. Once a bad run is cheap to undo, approving each step stops being worth it
3. **Add a never-delete rule to your `CLAUDE.md`.** Superseded files move to an archive folder, so a wrong step leaves nothing to recover
4. **Write the rules once, reuse them everywhere.** A global `CLAUDE.md` for every project, a project one for its conventions, `settings.local.json` for what should never prompt

If you remember one thing

- The rules you write matter more than the prompts you click
- Set up the guardrails first, then the permissions can be as loose as your work allows

Questions?

Happy to share the settings file or `CLAUDE.md` excerpts with anyone who wants them

These slides

idagri.github.io/files/ai-permissions-workshop-slides.pdf

My global `CLAUDE.md`, all nine rules

idagri.github.io/files/ida-claude-md.md

Idaliya Grigoryeva · idaliya.gri@gmail.com