PLAN

The Claude Code plan mode cheat sheet

Free guide. No signup. 1 min read

Checked 2026-10-23 against code.claude.com/docs/en/permission-modes.

From @learningwithcole

What plan mode does

Claude reads your files, runs shell commands to explore, and writes a plan. It does not edit your source. The edits wait until you approve the plan.

4 ways to turn it on

1. **Shift+Tab** until the status bar says `⏸ plan mode on`. From manual mode the cycle is: manual → accept edits → plan.
2. `/plan` + **your task** for a single prompt, e.g. `/plan add dark mode to the settings page`
3. **Start in it:** `claude --permission-mode plan`
4. **Make it the default** for a project: `"defaultMode": "plan"` under `permissions` in `.claude/settings.json`

Press Shift+Tab again to leave plan mode without approving anything.

When the plan is ready

- **Yes, and use auto mode**: approve and let it run (reads "Yes, auto-accept edits" if auto mode isn't available to you)
- **Yes, manually approve edits**: approve, then review each edit
- **No, keep planning**: stay in plan mode and tell Claude what to change
- **Ctrl+G**: open the plan in your own text editor and rewrite it before Claude starts

The one catch

In an interactive terminal session where bypass permissions are available, plan mode is not enforced. Claude is still told to plan without editing, but an edit it attempts will run without asking. Bypass is available when you start with `--dangerously-skip-permissions`, `--allow-dangerously-skip-permissions` or `--permission-mode bypassPermissions` (or set it as your default mode). If you need the guard rail, start without them.

3 copy-paste planning prompts

1. Scope it first

```javascript
/plan [your task]. Before the plan, list every file you expect to change and why. Flag anything outside that list as a separate step I can say no to.
```

2. Smallest safe change

```javascript
/plan [your task]. Give me the smallest change that works, in numbered steps. For each step, say how I can check it worked. No refactors I didn't ask for.
```

3. Ask before you assume

```javascript
/plan [your task]. Read the relevant code first, then ask me up to 3 questions about anything unclear before you write the plan.
```

Habit worth keeping

Commit before you approve a big plan: `git add -A && git commit -m "before plan"`. Then a bad run is one `git restore .` away.