The Claude Command Cheat Sheet

60+ commands, shortcuts, and workflow tricks for Claude Code — on two pages you can print and keep next to you.

Most people run Claude like a chat box: type a question, read the answer, repeat. But the people who get real work out of it are running commands, shortcuts, and modes you've probably never touched. Here's the whole surface, in one place.

> New to Claude Code? Install it once: `npm install -g @anthropic-ai/claude-code`, then run `claude` in any project folder. Commands change a little between versions — run `/help` in your session to see the exact list for yours.

---

1 · Keyboard shortcuts (inside a session)

Key	What it does
`Enter`	Send your message
`\` + `Enter` (or `Option/Alt + Enter`)	New line without sending — write a multi-line prompt
`Esc`	Interrupt Claude mid-answer / stop the current action
`Esc` `Esc` (double-tap)	Rewind — jump back and edit an earlier message
`Ctrl + C`	Cancel what you're typing (press twice to quit)
`Ctrl + D`	Exit the session
`Ctrl + L`	Clear the screen (keeps your context)
`Ctrl + R`	Toggle verbose output / expand the last tool result
`↑` / `↓`	Scroll back through your prompt history
`Shift + Tab`	Cycle modes: normal → auto-accept edits → **plan mode**
`Ctrl + V`	Paste an image from your clipboard straight into the prompt
Drag a file onto the window	Attach that file

2 · The four "magic keys" (type these at the start of a line)

| Symbol | What it does | |---|---| | `/` | Open the slash-command menu (see below) | | `@` | Mention a file or folder — `@src/app.ts` pulls it into context | | `!` | Bash mode — run a shell command directly and feed the output to Claude | | `#` | Save a note to memory — writes it into your `CLAUDE.md` for every future session |

---

3 · Slash commands (the control panel)

Session & context | Command | What it does | |---|---| | `/help` | List every command for your version | | `/clear` | Wipe the conversation and start fresh | | `/compact` | Summarize the chat to free up context without losing the thread | | `/context` | See how much of the context window you're using | | `/cost` | Show token spend for this session | | `/usage` | Your plan usage and limits | | `/status` | Session, account, and connection status | | `/resume` | Reopen a past conversation | | `/export` | Export the conversation | | `/rewind` | Roll the conversation (and file changes) back to an earlier point |

Setup & config | Command | What it does | |---|---| | `/init` | Scan the repo and generate a starter `CLAUDE.md` | | `/memory` | Open and edit your `CLAUDE.md` memory files | | `/config` | Open settings | | `/model` | Switch model (e.g. Opus / Sonnet) | | `/permissions` | View and edit tool permissions | | `/add-dir` | Give Claude another working directory | | `/terminal-setup` | Enable `Shift+Enter` for newlines in your terminal | | `/vim` | Vim keybindings in the input | | `/statusline` | Customize the status line | | `/output-style` | Change how responses are formatted | | `/login` · `/logout` | Switch or sign out of your account | | `/doctor` |

Diagnose your install | | `/bug` | Report a bug to Anthropic | | `/release-notes` | What changed in the latest version |

Power tools | Command | What it does | |---|---| | `/agents` | Create and manage subagents (specialized helpers) | | `/mcp` | Connect MCP servers (external tools & data) | | `/hooks` | Set hooks that run automatically on events | | `/plugin` | Add plugin marketplaces and install plugins | | `/review` | Review a pull request | | `/pr-comments` | Pull GitHub PR comments into the session | | `/ide` | Connect your IDE (VS Code / JetBrains) |

> Make your own: drop a markdown file in `.claude/commands/`, e.g. `ship.md`, and it becomes `/ship`. Reusable prompts on tap.

---

4 · Workflow tricks that actually change how it feels

- **Plan mode** (`Shift+Tab` to it): Claude reads and plans but touches nothing until you approve. Use it before any big change.
- **Auto-accept mode** (`Shift+Tab`): stop clicking "yes" on every edit when you're moving fast on trusted work.
- **Think harder**: the words `think` → `think hard` → `think harder` → `ultrathink` each give it more room to reason before acting. Use on the hard problems.
- `CLAUDE.md` : a file in your project Claude reads every session — put your stack, rules, and preferences there once and stop re-explaining yourself.
- `@` **a whole folder**: `@src/` pulls in the directory so it has the real context, not a guess.
- `!` **before a command**: run `!npm test` and Claude sees the output and reacts.

- `#` **to remember**: end a line you want kept with `#`, and it's saved to memory for good.
- **Headless mode**: `claude -p "..."` runs a one-shot from your terminal or a script — great for automations.

---

5 · Terminal commands (outside a session)

```
claude                      # start an interactive session in
this folder
claude "fix the failing test"   # start with a first prompt
claude -c                   # continue your most recent
conversation
claude -r                   # resume — pick a past session
claude -p "summarize this repo"  # headless: print the answer
and exit
claude --model opus         # choose the model
claude --add-dir ../shared  # include another directory
claude --output-format json # structured output (great with -p,
for scripts)
claude --dangerously-skip-permissions   # skip approval prompts
(use with care)
claude update               # update to the latest version
claude doctor               # health-check your setup
claude --version            # which version am I on
```

MCP (connect external tools & data):

```
claude mcp add <name> -- <command>   # add an MCP server
claude mcp list                      # see connected servers
claude mcp remove <name>             # remove one
```

---

6 · Skills & agents (the part most people never turn on)

- **Skills** — a `SKILL.md` (plus files) in `.claude/skills/<name>/` that teaches Claude one job really well; it loads automatically when relevant. Find thousands at **skills.sh** (install with `npx skills add <owner/repo>`).
- **Subagents** — spin up specialized helpers with `/agents` (a reviewer, a researcher, a test-writer) that work in their own context so your main thread stays clean.
- **Hooks** — with `/hooks`, run your own command automatically before/after a tool runs (e.g. auto-format after every edit, or block edits to a protected file).
- **Plugins** — `/plugin` adds whole bundles (commands + agents + skills + MCP) from a marketplace in one step.

The real unlock isn't any single command — it's stacking them. Plan mode to think, subagents to divide the work, skills for the repeatable jobs, hooks to enforce your rules, `CLAUDE.md` so it always knows your setup. That's when Claude stops feeling like a chatbot and starts working like it already knows the job.

Built by Axionox — I build AI systems and automations for a living, and this is the toolkit I actually use every day. Want more like this? Follow

along, and save this sheet so it's there when you need it.