

Claude Code / Cheatsheet

A quick reference to slash commands: what they do, when to use them, and how to combine them into a workflow.

Updated: July 2026 · Source: official Anthropic documentation (code.claude.com/docs) · Command availability varies by version, plan and platform — type `/help` to see the ones active for you.

How to use them

Type `/` at the start of a message: an autocomplete menu appears. Keep typing to filter.

Arguments

Text after the command becomes its argument: `/plan fix the auth bug`.
`<arg>` = required, `[arg]` = optional.

Skills and Workflows

Some commands are **skills** (prompts Claude can also invoke on its own) or **workflows** (fan-out across many subagents).

MCP prompts

Connected MCP servers expose prompts as commands in the form `/mcp_server_prompt`.

► The typical flow in 6 moves

- 1. First session in the repo:** `/init` generates the `CLAUDE.md` → `/memory` to refine it → `/mcp` for servers → `/permissions` for approval rules.
- 2. During the task:** `/plan` before big changes → `/model` and `/effort` to tune power and reasoning → `/btw` for quick questions.
- 3. Long conversation:** `/context` shows what's filling the window → `/compact` summarizes it to free up space.
- 4. Parallel work:** `/fork` to delegate to a subagent → `/background` to detach the session → `/batch` for large-scale changes in separate worktrees.
- 5. Before pushing:** `/diff` → `/code-review` (with `--fix` to apply) → `/security-review` for the security pass.
- 6. If something goes wrong:** `/rewind` restores code and chat to a checkpoint → `/doctor` diagnoses the installation.

⚙️ Setup and project

the first commands in a new repo

<code>/init</code>	Analyzes the project and generates a starter CLAUDE.md : structure, build commands, conventions. Run it first in every new repo.
<code>/memory</code>	Opens the CLAUDE.md memory files for editing; also manages auto-memory. What you write here persists across sessions.
<code>/permissions</code> alias: <code>/allowed-tools</code>	Manages the allow / ask / deny rules for tools: what Claude can do without asking, what needs confirmation, what is forbidden.
<code>/mcp [reconnect enable disable]</code>	Manages MCP servers: connections, OAuth, reconnection, enabling/disabling one or all of them.
<code>/config [key=value]</code> alias: <code>/settings</code>	Opens the settings (theme, model, output style...). Direct shortcut: <code>/config theme=dark</code> , <code>/config model=sonnet</code> .
<code>/add-dir <path></code>	Adds an extra working directory for file access in the current session.
<code>/cd <path></code>	Moves the session to another directory while preserving the prompt cache : the new <code>CLAUDE.md</code> is appended instead of rebuilding everything.
<code>/help</code>	Shows all commands available in your installation. The source of truth whenever you're unsure.

🔄 Context and session

managing the context window

/clear [name] alias: /reset, /new	New conversation with an empty context (project memory stays). The previous chat remains recoverable with /resume; the optional name labels it.
/compact [instructions]	Summarizes the conversation to free context without losing the thread. You can steer the summary: <i>/compact keep the error handling patterns</i> .
/context [all]	Displays context usage as a colored grid, with hints about heavy tools, bloated memory, and capacity warnings.
/btw <question>	Quick question outside the conversation : the answer doesn't enter the context. Works even while Claude is still responding.
/rewind alias: /checkpoint, /undo	Restores conversation and/or code to an earlier checkpoint. Shortcut: double Esc . You can choose to rewind only the code or only the chat.
/export [file]	Exports the conversation as text: to a file, or via a dialog to copy to the clipboard. Handy for documenting a problem you've just solved.
/copy [N]	Copies the last response (or the Nth) to the clipboard. With code blocks it opens a picker; w writes the selection to a file (handy over SSH).
/rename [name] / /recap	/rename names the session (or auto-generates a name); /recap produces a one-line summary of what has happened so far.

↔ Navigating between sessions

resume, branch, move around

/resume [session] alias: /continue	Resumes a conversation by ID or name, or opens the picker (background sessions are marked bg). From the CLI: claude -c resumes the latest one.
/branch [name]	Creates a conversation branch at the current point, like a git branch: try a different path without losing the original, recoverable with /resume.
/teleport alias: /tp	Pulls a Claude Code on the web session into your terminal: downloads branch and conversation.
/remote-control alias: /rc	Makes the local session controllable from claude.ai : keep going from your phone or another device.
/desktop alias: /app	Continues the current session in the Claude Code desktop app (macOS/Windows, subscription required).

Model, effort and costs

power vs. wallet

/model [model]	Switches model and saves it as the default. Without an argument it opens the picker; s on a row switches for the current session only. Beware: mid-conversation the history is re-read without cache.
/effort [level auto]	Adjusts the reasoning level : low, medium, high, xhigh, max, ultracode (xhigh + automatic workflow orchestration). Without an argument it opens an interactive slider. Takes effect immediately.
/fast [on off]	Toggles fast mode for quicker responses.
/usage alias: /cost, /stats	Session cost, plan limits and statistics, with a breakdown by skills, subagents, plugins and MCP servers.
/usage-credits	Configures extra credits to keep working when you hit a plan limit.

✓ Planning, review and quality

before breaking things and before pushing

/plan [description]	Enters plan mode : Claude proposes a complete plan (which files, why, in what order) before touching anything. Shortcut: Shift+Tab cycles through modes.
/goal [condition clear]	Sets a goal : Claude keeps working on it, turn after turn, until the condition is met. Without an argument it shows the active goal; <i>clear</i> removes it.
/diff	

	Interactive diff viewer: left/right arrows switch between the git diff and Claude's individual turns, up/down between files. Refreshes automatically when the git state changes.
<code>/code-review [level] [--fix] [--comment]</code> SKILL	Reviews the current diff looking for correctness bugs and cleanups. --fix applies the findings, --comment posts them as inline comments on the GitHub PR, ultra launches a multi-agent review in the cloud.
<code>/review [PR]</code>	Same review, but on a GitHub pull request (by number). Without arguments it lists open PRs to pick from. Read-only.
<code>/security-review</code>	Security pass over the branch diff: injection, auth issues, data exposure. Read-only.
<code>/simplify [target]</code> SKILL	Cleanup-only review with fixes applied: 4 agents in parallel covering reuse, simplification, efficiency and abstraction level. It doesn't hunt for bugs: that's what <code>/code-review</code> is for.
<code>/ultrareview [PR]</code> CLOUD	Deep multi-agent review in a cloud sandbox. Preferred invocation: /code-review ultra . 3 free runs on Pro/Max, then credits.
<code>/verify /run</code> SKILL	<code>/verify</code> builds, launches and observes the app to confirm the change actually works (not just that tests pass); <code>/run</code> starts and drives the project's app. /run-skill-generator teaches Claude how to do this for your project.
<code>/dataviz [request]</code> SKILL	Design guidance for charts and dashboards: picks the right form, validates the palette for contrast and color blindness.

⇒ Parallelism and background more work, same terminal	
<code>/fork <directive></code>	Launches a background subagent that inherits the whole conversation and works on the directive while you keep going. The result lands back in your chat when it's done.
<code>/background [prompt]</code> alias: <code>/bg</code>	Detaches the whole session into the background and frees the terminal. Monitor it with <i>claude agents</i> ; the optional prompt is the last instruction before detaching.
<code>/batch <instruction></code> SKILL	Large-scale changes: analyzes the codebase, splits the work into 5-30 independent units and, once the plan is approved, launches one subagent per unit in an isolated git worktree ; each one implements, tests and opens a PR.
<code>/tasks</code> alias: <code>/bashes</code>	Lists and manages everything running in the background in the current session.
<code>/loop [interval] [prompt]</code> SKILL	Runs a prompt repeatedly while the session stays open: <i>/loop 5m check whether the deploy has finished</i> . Without a prompt it does autonomous maintenance or uses <code>.claude/loop.md</code> .
<code>/schedule [description]</code> alias: <code>/routines</code>	Creates scheduled routines that run on Anthropic cloud infrastructure. Claude walks you through the setup conversationally.
<code>/workflows /stop</code>	<code>/workflows</code> opens the workflow progress view (pause/resume/save); <code>/stop</code> stops the background session you're attached to (to detach without stopping it: <code>/exit</code>).
<code>/deep-research <question></code> WORKFLOW	Fan-out web research: many searches in parallel, cross-checking of sources, and a final report with citations .

+ Integrations

GitHub, Slack, IDE, browser, cloud

<code>/install-github-app</code>	Installs the Claude GitHub App on a repo, with optional setup of GitHub Actions workflows and secrets.
<code>/autofix-pr</code> <code>[prompt]</code> CLOUD	Cloud session that watches the PR of the current branch and pushes fixes when CI fails or reviewer comments come in. Requires the <i>gh</i> CLI.
<code>/install-slack-app</code> / <code>/chrome</code>	Install/configure the Slack app (via OAuth) and Claude in Chrome.
<code>/ide</code>	Manages IDE integrations (VS Code, JetBrains...) and shows their status.
<code>/web-setup</code> / <code>/remote-env</code>	<code>/web-setup</code> connects GitHub to Claude Code on the web using your local <i>gh</i> credentials; <code>/remote-env</code> picks the default environment for cloud agents.
<code>/ultraplan</code> <code><prompt></code> CLOUD	Prepares a plan in an ultraplan session, you review it in the browser , then execute remotely or send it back to the terminal.
<code>/claude-api</code> <code>[migrate]</code> SKILL	Loads the Claude API reference for the project's language (tool use, streaming, batch...). migrate updates existing code to a newer model.

🔧 Extensions and customization

skills, plugins, hooks, subagents

<code>/skills</code> / <code>/reload-skills</code>	<code>/skills</code> lists the available skills (filter by typing, t sorts by tokens, Space toggles visibility); <code>/reload-skills</code> re-scans the directories without restarting.
<code>/plugin</code> <code>[list install enable disable]</code>	Manages plugins (bundles of commands, agents, skills and hooks distributed as Git repos). <code>/reload-plugins</code> reloads them on the fly.
<code>/hooks</code>	Displays the hook configurations: deterministic scripts attached to tool events (e.g. <code>PreToolUse</code> on Bash).
<code>/agents</code>	Subagent management: in recent versions you just ask Claude to create them, or edit the files in <code>.claude/agents/</code> .
<code>/statusline</code> / <code>/theme</code> / <code>/keybindings</code>	Customize the status line (describe what you want), the color theme (including colorblind variants and custom themes) and keyboard shortcuts.
<code>/color</code> / <code>/tui</code> <code>[fullscreen]</code> / <code>/focus</code>	<code>/color</code> changes the prompt bar color; <code>/tui</code> enables the flicker-free fullscreen renderer; <code>/focus</code> shows only the prompt, a tool summary and the final answer.
<code>/team-onboarding</code>	Generates an onboarding guide for teammates by analyzing your Claude Code usage over the last 30 days: sessions, commands, MCP servers.
<code>/fewer-permission-prompts</code> SKILL	Analyzes your transcripts and adds an allowlist of your most-used read-only tools to <code>settings.json</code> , reducing confirmation prompts .

⚙️ Diagnostics and maintenance

when something's off

<code>/doctor</code>	Diagnoses installation and settings with status icons. Press f to have Claude fix the issues it found.
<code>/debug</code> <code>[description]</code> SKILL	Enables debug logging for the session and analyzes the log to troubleshoot; the optional description focuses the analysis.
<code>/status</code> / <code>/release-notes</code>	<code>/status</code> shows version, model, account and connectivity (works even while Claude is responding); <code>/release-notes</code> opens the interactive changelog.
<code>/feedback</code> <code>[report]</code> alias: <code>/bug</code> , <code>/share</code>	Reports a bug or sends feedback to Anthropic, with the session context attached.
<code>/insights</code> / <code>/heapdump</code>	<code>/insights</code> generates a report on your sessions (areas, patterns, friction); <code>/heapdump</code> writes a JS heap snapshot to diagnose abnormal memory usage.
<code>/login</code> / <code>/logout</code> / <code>/exit</code>	Account authentication and leaving the CLI (<code>/exit</code> from an attached background session detaches it without stopping it). Alias for <code>/exit</code> : <code>/quit</code> .

<code>/powerup</code>	Interactive mini-lessons with animated demos to discover Claude Code's features.
<code>/radio</code>	Opens Claude FM , the lo-fi radio, in the browser. For coding with the right soundtrack.
<code>/stickers</code> / <code>/mobile</code> / <code>/passes</code>	Order Claude Code stickers, show the QR code for the mobile app, gift friends a free week (if eligible).
<code>/voice</code> <code>[hold tap off]</code>	Voice dictation, in hold-to-talk or tap-to-toggle mode.

🔗 Create your own commands (skills)

Any prompt you repeat often can become a command. The recommended format is a **skill**; files in `.claude/commands/` remain supported as a legacy format.

```
# .claude/skills/commit/SKILL.md
---
description: Create a conventional commit
allowed-tools: Bash(git add:*), Bash(git commit:*)
argument-hint: [message]
---
Analyze the diff and create a commit
in Conventional Commits style: $ARGUMENTS
```

- Invoke it with `/commit`; **\$ARGUMENTS** captures all text after the command (\$1, \$2... the individual arguments).
- `!'command'` in the file runs shell and injects the output into the prompt.
- Project skills: `.claude/skills/` (commit them to the repo!) · personal: `~/.claude/skills/`.
- Unlike the old commands, Claude can also invoke skills **on its own** when the description matches the task.

🖱️ Shortcuts to remember

- `Esc` interrupts the response in progress · `Esc Esc` opens the rewind menu.
- `Shift+Tab` cycles: normal → auto-accept → plan mode.
- `!` at the start of a line = bash mode: run shell commands and the output enters the context.
- `@` + path = explicit reference to a file (with tab-completion).
- `#` + text = quick addition to memory (*# always use single quotes in JS*).

🔖 Field-tested tips

- Run `/compact` proactively in long sessions, not when you're already at the limit.
- `/plan` before every big change: it costs one turn, it saves ten.
- Slash commands **consume tokens**: even `/compact` has a cost.
- Prefer sequential invocations over chaining commands in a single prompt.
- Commit team skills to the repo and treat them like code: reviews included.

Source: code.claude.com/docs/en/commands · Features evolve fast: when in doubt, `/help` and `/release-notes` are the truth for your version.