Claude Code / Cheatsheet

Bignami dei comandi slash: cosa fanno, quando usarli e come combinarli in un flusso di lavoro.

Aggiornato: luglio 2026 · Fonte: documentazione ufficiale Anthropic (code.claude.com/docs) · La disponibilità dei comandi varia in base a versione, piano e piattaforma — digita **/help** per vedere quelli attivi da te.

Come si usano
Digita `/` a inizio messaggio: appare il menu con autocompletamento. Continua a scrivere per filtrare.

Argomenti
Il testo dopo il comando diventa argomento: `/plan fix del bug auth`. `<arg>` = obbligatorio, `[arg]` = opzionale.

Skill e Workflow
Alcuni comandi sono **skill** (prompt che Claude può invocare anche da solo) o **workflow** (fan-out su molti subagenti).

Prompt MCP
I server MCP connessi espongono prompt come comandi nel formato `/mcp__server__prompt`.

▶ Il flusso tipico in 6 mosse

1. **Prima sessione nel repo:** `/init` genera il CLAUDE.md → `/memory` per rifinirlo → `/mcp` per i server → `/permissions` per le regole di approvazione.
2. **Durante il task:** `/plan` prima di modifiche grosse → `/model` e `/effort` per regolare potenza e ragionamento → `/btw` per domande al volo.
3. **Conversazione lunga:** `/context` mostra cosa riempie la finestra → `/compact` la riassume liberando spazio.
4. **Lavoro in parallelo:** `/fork` per delegare a un subagente → `/background` per staccare la sessione → `/batch` per modifiche massicce in worktree separati.
5. **Prima di pushare:** `/diff` → `/code-review` (con `--fix` per applicare) → `/security-review` per la passata di sicurezza.
6. **Se qualcosa va storto:** `/rewind` riporta codice e chat a un checkpoint → `/doctor` diagnostica l'installazione.

⚙ Setup e progetto

i primi comandi in un repo nuovo

`/init`	Analizza il progetto e genera un **CLAUDE.md** di partenza: struttura, comandi di build, convenzioni. Da lanciare per primo in ogni repo nuovo.
`/memory`	Apre i file di memoria **CLAUDE.md** per modificarli; gestisce anche l'auto-memory. Ciò che scrivi qui persiste tra le sessioni.
`/permissions` `alias: /allowed-tools`	Gestisce le regole **allow / ask / deny** per i tool: cosa Claude può fare senza chiedere, cosa richiede conferma, cosa è vietato.
`/mcp` `[reconnect|enable|disable]`	Gestisce i server **MCP**: connessioni, OAuth, riconnessione, abilitazione/disabilitazione singola o di tutti.
`/config` `[chiave=valore]` `alias: /settings`	Apre le impostazioni (tema, modello, output style...). Scorciatoia diretta: **/config theme=dark**, **/config model=sonnet**.
`/add-dir` `<path>`	Aggiunge una directory di lavoro extra per l'accesso ai file nella sessione corrente.
`/cd` `<path>`	Sposta la sessione in un'altra directory **preservando la cache** del prompt: il CLAUDE.md nuovo viene accodato invece di ricostruire tutto.
`/help`	Mostra tutti i comandi disponibili nella tua installazione. La fonte di verità quando hai un dubbio.

↻ Contesto e sessione

gestire la finestra di contesto

/clear [nome] alias: /reset, /new	Nuova conversazione a contesto vuoto (la memoria di progetto resta). La chat precedente rimane recuperabile con /resume; il nome opzionale la etichetta.
/compact [istruzioni]	**Riassume la conversazione** per liberare contesto senza perdere il filo. Puoi indirizzare il riassunto: */compact mantieni i pattern di error handling*.
/context [all]	Visualizza l'uso del contesto come griglia colorata, con suggerimenti su tool pesanti, memoria gonfia e avvisi di capacità.
/btw <domanda>	Domanda al volo **fuori dalla conversazione**: la risposta non entra nel contesto. Funziona anche mentre Claude sta ancora rispondendo.
/rewind alias: /checkpoint, /undo	Riporta conversazione e/o **codice** a un checkpoint precedente. Scorciatoia: doppio **Esc**. Si può scegliere di riavvolgere solo il codice o solo la chat.
/export [file]	Esporta la conversazione come testo: su file, o via dialog per copiare negli appunti. Utile per documentare un problema appena risolto.
/copy [N]	Copia negli appunti l'ultima risposta (o la N-esima). Con blocchi di codice apre un picker; **w** scrive la selezione su file (comodo via SSH).
/rename [nome] / **/recap**	/rename dà un nome alla sessione (o lo autogenera); /recap produce un riassunto in una riga di quello che è successo finora.

⇄ Navigare tra le sessioni

riprendere, ramificare, spostarsi

/resume [sessione] alias: /continue	Riprende una conversazione per ID o nome, o apre il picker (le sessioni in background sono marcate **bg**). Da CLI: **claude -c** riprende l'ultima.
/branch [nome]	Crea un **ramo della conversazione** al punto attuale, come un git branch: provi una strada diversa senza perdere l'originale, recuperabile con /resume.
/teleport alias: /tp	Tira dentro il terminale una sessione di **Claude Code on the web**: scarica branch e conversazione.
/remote-control alias: /rc	Rende la sessione locale controllabile **da claude.ai**: continui dal telefono o da un altro dispositivo.
/desktop alias: /app	Continua la sessione corrente nell'app desktop di Claude Code (macOS/ Windows, con abbonamento).

Modello, effort e costi

potenza vs. portafoglio

/model [modello]	Cambia modello e lo salva come default. Senza argomento apre il picker; **s** su una riga cambia solo per la sessione corrente. Attenzione: con conversazione avviata la history viene riletta senza cache.
/effort [livello|auto]	Regola il **livello di ragionamento**: low, medium, high, xhigh, max, ultracode (xhigh + orchestrazione automatica di workflow). Senza argomento apre uno slider interattivo. Ha effetto immediato.
/fast [on|off]	Attiva/disattiva la **fast mode** per risposte più rapide.
/usage alias: /cost, /stats	Costo della sessione, limiti del piano e statistiche, con breakdown per skill, subagenti, plugin e server MCP.
/usage-credits	Configura i crediti extra per continuare a lavorare quando raggiungi un limite del piano.

✓ Pianificazione, review e qualità

prima di fare danni e prima di pushare

/plan [descrizione]	Entra in **plan mode**: Claude propone un piano completo (quali file, perché, in che ordine) prima di toccare qualsiasi cosa. Scorciatoia: **Shift+Tab** cicla le modalità.
/goal [condizione|clear]	Fissa un **obiettivo**: Claude continua a lavorarci finché la condizione non è soddisfatta, turno dopo turno. Senza argomento mostra il goal attivo; *clear* lo rimuove.

/diff	Viewer interattivo dei diff: frecce sinistra/destra per passare dal diff git ai singoli turni di Claude, su/giù per i file. Si aggiorna da solo se lo stato git cambia.
/code-review [livello] [--fix] [--comment] SKILL	Rivede il diff corrente cercando **bug di correttezza** e pulizie. **--fix** applica i finding, **--comment** li posta come commenti inline sulla PR GitHub, **ultra** lancia una review multi-agente nel cloud.
/review [PR]	Stessa review, ma su una **pull request GitHub** (per numero). Senza argomenti elenca le PR aperte tra cui scegliere. Sola lettura.
/security-review	Passata di **sicurezza** sul diff del branch: injection, problemi di auth, esposizione di dati. Sola lettura.
/simplify [target] SKILL	Review di **solo cleanup** con fix applicati: 4 agenti in parallelo su riuso, semplificazione, efficienza e livello di astrazione. Non cerca bug: per quelli c'è /code-review.
/ultrareview [PR] CLOUD	Review profonda multi-agente in sandbox cloud. Invocazione preferita: **/code-review ultra**. 3 run gratuiti su Pro/Max, poi crediti.
/verify / **/run** SKILL	/verify **compila, lancia e osserva** l'app per confermare che la modifica funzioni davvero (non solo che i test passino); /run avvia e pilota l'app del progetto. **/run-skill-generator** insegna a Claude come farlo per il tuo progetto.
/dataviz [richiesta] SKILL	Linee guida di design per grafici e dashboard: sceglie la forma giusta, valida la palette per contrasto e daltonismo.

⇉ Parallelismo e background

più lavoro, stesso terminale

/fork <direttiva>	Lancia un **subagente in background** che eredita l'intera conversazione e lavora sulla direttiva mentre tu continui. Il risultato torna nella tua chat quando ha finito.
/background [prompt] alias: /bg	**Stacca l'intera sessione** in background e libera il terminale. Monitori con *claude agents*; il prompt opzionale è l'ultima istruzione prima del distacco.
/batch <istruzione> SKILL	Modifiche **su larga scala**: analizza il codebase, spezza il lavoro in 5-30 unità indipendenti e, approvato il piano, lancia un subagente per unità in un **git worktree** isolato; ognuno implementa, testa e apre una PR.
/tasks alias: /bashes	Elenca e gestisce tutto ciò che gira in background nella sessione corrente.
/loop [intervallo] [prompt] SKILL	Esegue un prompt **ripetutamente** finché la sessione è aperta: */loop 5m controlla se il deploy è finito*. Senza prompt fa manutenzione autonoma o usa .claude/loop.md.
/schedule [descrizione] alias: /routines	Crea **routine pianificate** che girano su infrastruttura cloud Anthropic. Claude ti guida nella configurazione in modo conversazionale.
/workflows / **/stop**	/workflows apre la vista di avanzamento dei workflow (pausa/riprendi/salva); /stop ferma la sessione in background a cui sei attaccato (per staccarti senza fermarla: /exit).
/deep-research <domanda> WORKFLOW	Ricerca web a ventaglio: molte ricerche in parallelo, verifica incrociata delle fonti e report finale **con citazioni**.

+ Integrazioni

GitHub, Slack, IDE, browser, cloud

/install-github-app	Installa la **Claude GitHub App** su un repo, con setup opzionale di workflow GitHub Actions e secrets.
/autofix-pr [prompt] CLOUD	Sessione cloud che **sorveglia la PR** del branch corrente e pusha fix quando la CI fallisce o arrivano commenti dei reviewer. Richiede la CLI *gh*.
/install-slack-app / **/chrome**	Installano/configurano l'app Slack (via OAuth) e Claude in Chrome.
/ide	Gestisce le integrazioni IDE (VS Code, JetBrains...) e ne mostra lo stato.
/web-setup / **/remote-env**	/web-setup collega GitHub a Claude Code on the web usando le credenziali gh locali; /remote-env sceglie l'ambiente di default per gli agenti cloud.
/ultraplan <prompt> CLOUD	Prepara un piano in una sessione ultraplan, lo rivedi **nel browser**, poi esegui in remoto o rimandi al terminale.
/claude-api [migrate] SKILL	Carica la reference della **Claude API** per il linguaggio del progetto (tool use, streaming, batch...). **migrate** aggiorna il codice esistente a un modello più recente.

✎ Estensioni e personalizzazione

skills, plugin, hook, subagenti

/skills / **/reload-skills**	/skills elenca le skill disponibili (filtro digitando, **t** ordina per token, **Spazio** cambia visibilità); /reload-skills ri-scansiona le directory senza riavviare.
/plugin [list|install|enable|disable]	Gestisce i **plugin** (bundle di comandi, agenti, skill e hook distribuiti come repo Git). /reload-plugins li ricarica al volo.
/hooks	Visualizza le configurazioni degli **hook**: script deterministici agganciati agli eventi dei tool (es. PreToolUse su Bash).
/agents	Gestione dei **subagenti**: nelle versioni recenti si chiede direttamente a Claude di crearli, o si editano i file in .claude/agents/.
/statusline / **/theme** / **/keybindings**	Personalizzano status line (descrivi cosa vuoi), tema colori (incluse varianti per daltonici e temi custom) e scorciatoie da tastiera.
/color / **/tui** [fullscreen] / **/focus**	/color cambia il colore della barra prompt; /tui attiva il renderer fullscreen senza sfarfallio; /focus mostra solo prompt, riepilogo tool e risposta finale.
/team-onboarding	Genera una **guida di onboarding** per i colleghi analizzando il tuo uso di Claude Code negli ultimi 30 giorni: sessioni, comandi, server MCP.
/fewer-permission-prompts SKILL	Analizza i tuoi transcript e aggiunge a settings.json una allowlist dei tool read-only più usati, **riducendo i prompt di conferma**.

⚒ Diagnostica e manutenzione

quando qualcosa non torna

/doctor	Diagnostica installazione e impostazioni con icone di stato. Premi **f** per far riparare a Claude i problemi trovati.
/debug [descrizione] SKILL	Attiva il **debug logging** per la sessione e analizza il log per risolvere problemi; la descrizione opzionale focalizza l'analisi.
/status / **/release-notes**	/status mostra versione, modello, account e connettività (funziona anche mentre Claude risponde); /release-notes apre il changelog interattivo.
/feedback [report] alias: /bug, /share	Segnala un bug o invia feedback ad Anthropic, con il contesto della sessione allegato.
/insights / **/heapdump**	/insights genera un report sulle tue sessioni (aree, pattern, attriti); /heapdump scrive uno snapshot della heap JS per diagnosticare consumi di memoria anomali.
/login / **/logout** / **/exit**	Autenticazione account e uscita dalla CLI (/exit da una sessione background attaccata la **stacca** senza fermarla). Alias di /exit: /quit.

★ Chicche	perché no
/powerup	Mini-lezioni interattive con demo animate per scoprire le funzionalità di Claude Code.
/radio	Apre **Claude FM**, la radio lo-fi, nel browser. Per programmare con la colonna sonora giusta.
/stickers / **/mobile** / **/passes**	Ordina gli sticker di Claude Code, mostra il QR per l'app mobile, regala una settimana gratis agli amici (se idoneo).
/voice `[hold|tap|off]`	Dettatura vocale, in modalità tieni-premuto o tocca per attivare.

✎ Crea i tuoi comandi (skills)

Ogni prompt che ripeti spesso può diventare un comando. Il formato raccomandato è una **skill**; i file in `.claude/commands/` restano supportati come formato legacy.

```
# .claude/skills/commit/SKILL.md
---
description: Crea un commit convenzionale
allowed-tools: Bash(git add:*), Bash(git commit:*)
argument-hint: [messaggio]
---
Analizza il diff e crea un commit
in stile Conventional Commits: $ARGUMENTS
```

- Si invoca con `/commit`; **$ARGUMENTS** cattura tutto il testo dopo il comando ($1, $2... i singoli argomenti).
- `!`comando`` nel file esegue shell e inietta l'output nel prompt.
- Skill di progetto: `.claude/skills/` (committale nel repo!) · personali: `~/.claude/skills/`.
- A differenza dei vecchi comandi, Claude può invocare le skill **anche da solo** quando la description corrisponde al task.

⌨ Scorciatoie da ricordare

- `Esc` interrompe la risposta in corso · `Esc Esc` apre il menu di rewind.
- `Shift+Tab` cicla: normale → auto-accept → plan mode.
- `!` a inizio riga = bash mode: esegui comandi shell e l'output entra nel contesto.
- `@` + percorso = riferimento esplicito a un file (con tab-completion).
- `#` + testo = aggiunta rapida alla memoria (*# usa sempre singoli apici in JS*).

✦ Consigli dal campo

- Lancia `/compact` in modo proattivo nelle sessioni lunghe, non quando sei già al limite.
- `/plan` prima di ogni modifica grossa: costa un turno, risparmia dieci.
- I comandi slash **consumano token**: anche /compact ha un costo.
- Meglio invocazioni sequenziali che catene di comandi in un unico prompt.
- Committa le skill di team nel repo e trattale come codice: review incluse.

Fonte: code.claude.com/docs/en/commands · Le funzionalità evolvono in fretta: in caso di dubbio, /help e /release-notes sono la verità della tua versione.