Lavoro parallelo e cloud ◆

Subagenti, agent view, batch su worktrees, agent teams, sessioni web e remote control: come far lavorare tre Claude mentre tu fai altro.

Aggiornato: luglio 2026 · Fonte: documentazione ufficiale Anthropic (code.claude.com/docs) · Vol. 6 della serie bignami Claude Code.

1. La scala del parallelismo

quattro livelli, dal più leggero al più pesante

Strumento	Cos'è	Quando	Costo relativo
Subagent	Worker dentro la tua sessione, contesto proprio, riporta solo il sunto al chiamante.	Task focalizzato dove conta solo il risultato.	Basso (risultati riassunti)
Background session (agent view)	Sessioni Claude Code complete e indipendenti, senza terminale attaccato, gestite da un supervisor.	Più task indipendenti in parallelo, tu supervisioni.	Ognuna consuma quota per conto suo
/batch (workflow)	Decompone un lavoro grosso in 5-30 unità, un subagente per unità in un worktree isolato, una PR ciascuno.	Modifica meccanica su tutto il codebase (migrazioni, rename).	Proporzionale alle unità
Agent team (sperimentale)	Sessioni indipendenti che si **messaggiano tra loro**, task list condivisa, auto-coordinamento.	Lavoro che richiede discussione: ipotesi in competizione, review parallela, feature con ownership separate.	Alto (~7× in plan mode)

2. Agent view

claude agents: la plancia di controllo

Un unico schermo per tutte le sessioni in background: cosa gira, cosa ha bisogno di te, cosa è finito. Ogni sessione è una conversazione Claude Code completa che continua **senza terminale attaccato** — un processo supervisor le ospita, sopravvivono alla chiusura della shell e allo sleep della macchina.

Il loop operativo

1. `claude agents` apre la vista; ogni prompt che digiti in basso **lancia una nuova sessione** (non un follow-up).
2. `Spazio` su una riga = **peek**: ultimo output o la domanda in attesa; rispondi da lì, con i tasti numerici per le scelte multiple, `Tab` per una risposta suggerita, `!` per mandare un comando Bash.
3. `Invio`/`→` = **attach**: la sessione prende il terminale, con recap di ciò che è successo; `←` su prompt vuoto = detach.
4. Da una sessione normale, `/bg` la stacca in background e la fa comparire come riga.

Leggere le righe

- Stati: animato = al lavoro, giallo = serve input, verde = completato, rosso = fallito, grigio = fermato; `•` = processo uscito ma riattaccabile (riparte da dov'era), `✣` = /loop che dorme tra le iterazioni.
- Il riassunto di riga è generato da un modello classe Haiku (refresh max ogni 15s); `2/5` = elementi paralleli completati/totali.
- **PR #N** a destra = la pull request aperta dalla sessione, colorata per stato: gialla in attesa di check/review, verde pronta, viola merged. Per molti task è lì che raccogli il risultato.
- Dalla shell: `claude attach/logs/stop/respawn/rm <id>`, e `claude agents --json` per lo scripting.

3. Batch, worktrees e fork

parallelismo senza pestarsi i piedi

/batch: la modifica su larga scala

`/batch migra src/ da Solid a React`: Claude studia il codebase, decompone in **5-30 unità indipendenti** e presenta un piano. Approvato, lancia un subagente per unità, ognuno in un **git worktree isolato**: implementa, testa, apre una PR. Tu fai il review di N pull request invece di sorvegliare N agenti. Richiede un repo git.

/fork vs /branch (facile confonderli)

- `/fork <direttiva>` = **delega**: spawna un subagente in background che eredita l'intera conversazione, lavora sulla direttiva mentre tu continui, e il risultato ti rientra in chat.
- `/branch [nome]` = **biforca te stesso**: copia della conversazione al punto attuale, tu passi nel ramo, l'originale resta recuperabile con /resume. Il git branch delle conversazioni.

Worktrees: perché sono la chiave

Un worktree è una copia di lavoro separata dello stesso repo: ogni agente edita la sua, zero conflitti con il tuo albero. Li usano /batch, i subagenti con `isolation: worktree` (worktree temporaneo, ripulito se non ci sono modifiche) e il flag `--worktree` di lancio. Vivono sotto `.claude/worktrees/`.

/tasks e /workflows

`/tasks` elenca tutto ciò che gira in background nella sessione corrente; `/workflows` apre la vista di avanzamento dei workflow dinamici (pausa, riprendi, salva). `/background` stacca l'intera sessione e libera il terminale.

4. Cloud e multi-dispositivo

il terminale non è più un confine

Strumento	Cosa fa
`Claude Code on the web`	Sessioni in sandbox cloud Anthropic dal browser o dal telefono: colleghi un repo GitHub, dai il task, fai review della PR. Ambienti configurabili (setup script, accesso rete, Docker); `/remote-env` sceglie il default.
`/teleport (/tp)`	Tira una sessione web **dentro il terminale**: scarica branch e conversazione e continui in locale.
`/remote-control (/rc)`	L'inverso: la sessione **locale** diventa controllabile da claude.ai o dall'app mobile. Il classico "continuo dal divano". Esiste anche in modalità server: `claude remote-control`.
`/ultraplan <prompt>`	Prepara il piano in una sessione cloud, lo rivedi nel browser, poi esegui in remoto o lo rimandi al terminale.
`/autofix-pr [prompt]`	Sessione cloud che sorveglia la PR del branch corrente e pusha fix quando la CI fallisce o arrivano commenti dei reviewer.
`/schedule (/routines)`	Routine su infrastruttura gestita: a orario, via API o su eventi GitHub. Il cron che non richiede una macchina accesa.
`/desktop / /mobile`	Continua nell'app desktop (sessioni parallele con isolamento git, pannelli, diff visuale) / QR per l'app mobile con notifiche push quando Claude ha bisogno di te.

5. Agent teams

sperimentale: attiva solo se ti serve davvero

Si abilitano con `CLAUDE_CODE_EXPERIMENTAL_AGENT_TEAMS=1` (env o settings.json). Ogni teammate è una sessione Claude Code indipendente; comunicano con messaggi diretti e una task list condivisa, e possono riusare le definizioni dei tuoi subagenti (tools e model dal frontmatter, corpo come istruzioni aggiuntive).

▶ Regole d'oro per non farsi male (col portafoglio)

- **Sonnet per i teammate**: bilancia capacità e costo per il coordinamento.
- **Team piccoli**: il consumo è ~proporzionale al numero di teammate; in plan mode si arriva a ~7× una sessione normale.
- **Prompt di spawn asciutti**: CLAUDE.md, MCP e skill si caricano da soli; tutto ciò che aggiungi entra nel contesto di ognuno.
- **Spegni chi ha finito**: un teammate attivo consuma finché non esce.
- **Il punto di transizione**: passa ai team solo quando i subagenti paralleli sbattono sui limiti di contesto o avrebbero bisogno di parlarsi. Per tutto il resto, subagenti e agent view bastano e costano meno.

Fonti: code.claude.com/docs — agents, agent-view, agent-teams, claude-code-on-the-web, remote-control, routines · Molte di queste feature sono in research preview: /release-notes per lo stato attuale.