Parallel work and cloud ◆

Subagents, agent view, batches on worktrees, agent teams, web sessions and remote control: how to keep three Claudes working while you do something else.

Updated: July 2026 · Source: official Anthropic documentation (code.claude.com/docs) · Vol. 6 of the Claude Code cheatsheet series.

1. The parallelism ladder

four levels, from lightest to heaviest

Tool	What it is	When	Relative cost
Subagent	Worker inside your session, own context, reports only a summary back to the caller.	Focused task where only the result matters.	Low (summarized results)
Background session (agent view)	Complete, independent Claude Code sessions with no terminal attached, managed by a supervisor.	Several independent tasks in parallel, you supervise.	Each consumes quota on its own
/batch (workflow)	Decomposes a large job into 5-30 units, one subagent per unit in an isolated worktree, one PR each.	Mechanical changes across the whole codebase (migrations, renames).	Proportional to the units
Agent team (experimental)	Independent sessions that **message each other**, shared task list, self-coordination.	Work that requires discussion: competing hypotheses, parallel review, features with separate ownership.	High (~7× in plan mode)

2. Agent view

claude agents: the control panel

A single screen for all background sessions: what is running, what needs you, what is done. Each session is a complete Claude Code conversation that keeps going **with no terminal attached** — a supervisor process hosts them, so they survive closing the shell and the machine going to sleep.

The operating loop

1. `claude agents` opens the view; every prompt you type at the bottom **launches a new session** (not a follow-up).
2. `Space` on a row = **peek**: latest output or the pending question; answer from there, with number keys for multiple choice, `Tab` for a suggested reply, `!` to send a Bash command.
3. `Enter`/`→` = **attach**: the session takes over the terminal, with a recap of what happened; `←` on an empty prompt = detach.
4. From a normal session, `/bg` detaches it into the background and makes it appear as a row.

Reading the rows

- States: animated = working, yellow = needs input, green = completed, red = failed, gray = stopped; `•` = process exited but re-attachable (resumes where it was), `✣` = / loop sleeping between iterations.
- The row summary is generated by a Haiku-class model (refresh at most every 15s); `2/5` = parallel items completed/total.
- **PR #N** on the right = the pull request opened by the session, colored by state: yellow awaiting checks/review, green ready, purple merged. For many tasks that is where you collect the result.
- From the shell: `claude attach/logs/stop/respawn/rm <id>`, and `claude agents --json` for scripting.

3. Batch, worktrees and fork

parallelism without stepping on each other's toes

/batch: changes at scale

`/batch migrate src/ from Solid to React`: Claude studies the codebase, decomposes it into **5-30 independent units** and presents a plan. Once approved, it launches one subagent per unit, each in an **isolated git worktree**: implement, test, open a PR. You review N pull requests instead of babysitting N agents. Requires a git repo.

/fork vs /branch (easy to confuse)

- `/fork <directive>` = **delegate**: spawns a background subagent that inherits the entire conversation, works on the directive while you keep going, and the result comes back into your chat.
- `/branch [name]` = **fork yourself**: a copy of the conversation at the current point, you move into the branch, the original stays recoverable with /resume. The git branch of conversations.

Worktrees: why they are the key

A worktree is a separate working copy of the same repo: each agent edits its own, zero conflicts with your tree. They are used by /batch, by subagents with `isolation: worktree` (temporary worktree, cleaned up if there are no changes) and by the `--worktree` launch flag. They live under `.claude/worktrees/`.

/tasks and /workflows

`/tasks` lists everything running in the background in the current session; `/workflows` opens the progress view for dynamic workflows (pause, resume, save). `/background` detaches the entire session and frees the terminal.

4. Cloud and multi-device

the terminal is no longer a boundary

Tool	What it does
`Claude Code on the web`	Sessions in an Anthropic cloud sandbox from the browser or your phone: connect a GitHub repo, give it the task, review the PR. Configurable environments (setup script, network access, Docker); `/remote-env` picks the default.
`/teleport (/tp)`	Pulls a web session **into the terminal**: downloads branch and conversation and you continue locally.
`/remote-control (/rc)`	The reverse: the **local** session becomes controllable from claude.ai or the mobile app. The classic "keep going from the couch". Also available in server mode: `claude remote-control`.
`/ultraplan <prompt>`	Prepares the plan in a cloud session, you review it in the browser, then execute remotely or send it back to the terminal.
`/autofix-pr [prompt]`	Cloud session that watches the current branch's PR and pushes fixes when CI fails or reviewer comments come in.
`/schedule (/routines)`	Routines on managed infrastructure: on a schedule, via API or on GitHub events. The cron that doesn't need a machine left on.
`/desktop / /mobile`	Continue in the desktop app (parallel sessions with git isolation, panels, visual diff) / QR code for the mobile app with push notifications when Claude needs you.

5. Agent teams

experimental: enable only if you really need it

Enabled with `CLAUDE_CODE_EXPERIMENTAL_AGENT_TEAMS=1` (env or settings.json). Each teammate is an independent Claude Code session; they communicate through direct messages and a shared task list, and can reuse your subagent definitions (tools and model from the frontmatter, body as extra instructions).

▶ Golden rules to avoid getting hurt (in the wallet)

- **Sonnet for teammates**: balances capability and cost for coordination.
- **Small teams**: consumption is roughly proportional to the number of teammates; in plan mode it can reach ~7× a normal session.
- **Lean spawn prompts**: CLAUDE.md, MCP and skills load on their own; everything you add goes into each teammate's context.
- **Shut down whoever is done**: an active teammate keeps consuming until it exits.
- **The transition point**: move to teams only when parallel subagents hit context limits or would need to talk to each other. For everything else, subagents and agent view are enough and cost less.

Sources: code.claude.com/docs — agents, agent-view, agent-teams, claude-code-on-the-web, remote-control, routines · Many of these features are in research preview: /release-notes for current status.