EcomLabs

PRACTICAL GUIDE

Claude Code: four habits to make your usage limit last, and a command cheat sheet

Why every message costs more than the one before, the four habits from the docs that stretch your usage limit (/clear, /compact, short sessions and the right model for the task) and a cheat sheet of the commands that show how much context you are carrying.

FOR	Anyone using Claude Code on a subscription plan who runs out of limit midway through the day
TESTED WITH	Only the example's arithmetic, on Node 22; the commands come from the docs and were not run in a fresh session
UPDATED	November 12, 2026
DOCS	code.claude.com/docs/en/costs
ARTICLE	ecomlabs.dev/g/tokens-en
AUTHOR	Luis Morales · @luismoralesdev

Contents

Before you start	Why your usage limit runs out
Step 01	The four habits
Step 02	Command cheat sheet
When you are done	Checklist

BEFORE YOU START

Why your usage limit runs out

Claude Code makes a new request with every message and the model remembers nothing between requests, so it resends the system prompt, the project context and every earlier message and tool result. Caching makes that resend cheaper, but per the costs page of the docs, a one-line question in a session that has been open all day still draws usage for the whole conversation.

Illustrative example, an arithmetic model computed with Node 22 (15,000 tokens of base context plus 3,000 per turn, one request per turn, no caching): a 20-turn session sends 930,000 tokens and two 10-turn sessions send 630,000, which is 32% less for the same work. It does not measure your plan: the docs publish no quotas in tokens, only a five-hour session limit and a weekly one on Pro and Max.

STEP 01

The four habits

HABIT	WHEN	WHAT IT DOES OR COSTS
`/clear`	When you move to unrelated work	Empty context; costs nothing. Come back with `/resume`
`/compact [instructions]`	A natural break in the same task	Summarizes the history; a large request. Warm cache: a fraction; after a long break it reprocesses everything
Short sessions	One task per session	Less context per request. The cache lasts one hour on a subscription within plan usage, five minutes with credits, API or cloud
`/model`	At the start of the session	Each model has its own cache: switching mid-session reads everything uncached. Sonnet for most work, Opus for the complex

- **Compact with a focus**: in the `CLAUDE.md` at your project root, a `# Compact instructions` section with `When you are using compact, please focus on test output and code changes`, the docs' text.
- **Coming back**: `/resume` sends the whole conversation again; if the cache expired, you pay for it with no discount.

> **CAUTION**
>
> `/model name` saves it as the default for new sessions. To change only this session, open the picker with `/model` and press `s` on the row.

STEP 02

Command cheat sheet

COMMAND	WHEN	WHAT IT DOES
`/context`	Before a long task	Grid of context usage, with suggestions
`/usage`	Midway through the day	Plan bars and a breakdown by skills, subagents, plugins and MCP; aliases `/cost`, `/stats`
`/rename [name]`	Before `/clear`	Names the session so you can resume it
`/clear`	When you switch tasks	New conversation (aliases `/reset`, `/new`)
`/compact [instructions]`	A break in the same task	Summarizes; accepts focus instructions
`/model`	At the start	Switches the model; with no argument it opens the picker

A routine for the day, exactly as you would type it in the session:

```
/context
/usage
/rename cart-discounts
/clear
/model
/compact keep the changes in components/cart/ and the test commands
```

WHEN YOU ARE DONE

Checklist

- [ ] One task per session; `/rename` and `/clear` when you finish.
- [ ] `/compact` only at natural breaks, with focus instructions.
- [ ] Model and effort chosen at the start, not mid-task.
- [ ] `/context` before a long task and `/usage` midway through the day.
- [ ] You do not resume an old session if you can start clean.
- [ ] Scheduled tasks stopped when you finish: they fire even while the session is idle.