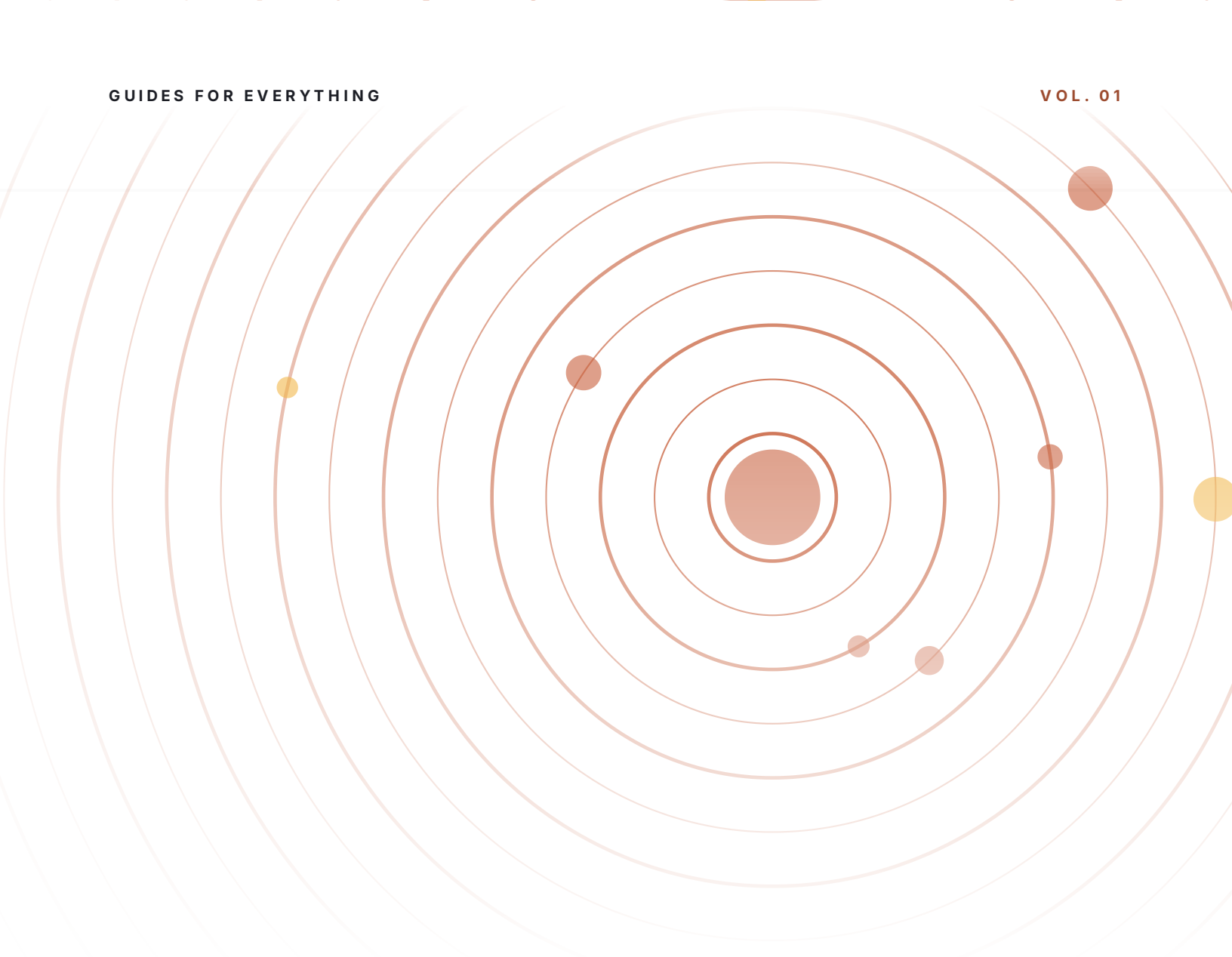




A BEGINNER'S MANUAL

GROK EDITION

Claude Code

Your first AI coding partner, from installation to everyday mastery

Claude Code

Your first AI coding partner, from installation to everyday mastery

GROK EDITION

Concise edition, revised and fact-checked by Grok Bot in GrokAllManualsV1.

Claude Code is an AI assistant from Anthropic that can read your project, write and edit files, run commands, and explain everything it does in plain English. You talk to it the way you'd talk to a helpful colleague, and it does the work.

This manual assumes you have **never used a terminal before**. Each chapter builds on the one before it. By the end you'll have installed Claude Code, built and changed real files with it, learned to control what it is allowed to do, and set it up to remember how you like to work.

HOW TO READ THIS MANUAL

Tip — a shortcut or good habit.

Warning — easy to get wrong.

`$ command` — text you type, exactly as shown.

Note — useful background.

Danger — can lose data, money, or access.

`Settings > Privacy` — a menu path to click.

New words appear in **bold** the first time and are collected in the Glossary at the back.

Contents

01 Welcome to Claude Code	6
1.1 What is an “agent”?	6
1.2 What can you do with it?	7
1.3 Where Claude Code runs	7
1.4 What you need	8
02 Terminal Basics in Ten Minutes	9
2.1 Opening a terminal	9
2.2 Folders and paths	10
2.3 The five commands you need	10
2.4 Copying and pasting in a terminal	11
03 Install and Sign In	12
3.1 Install with the native installer	12
3.2 Choose a plan	14
3.3 Log in for the first time	14
3.4 Health check and updates	15
04 Your First Session	16
4.1 The Claude Code screen	16
4.2 Build a small website	17
4.3 Ask questions about existing code	17
4.4 Approving, interrupting, and undoing	18
4.5 Ending and resuming	18
05 Talking to Claude Effectively	19
5.1 Five habits of great prompts	19
5.2 Plan mode: look before you leap	20
5.3 Share what you see	20
5.4 Ask a side question	20
5.5 Keep conversations focused	21
06 Permissions and Safety	22
6.1 Permission modes	22

6.2	Allow, ask, and deny rules	23
6.3	Where settings live	24
6.4	Safety nets	24
07	Memory: Teaching Claude Your Project	25
7.1	CLAUDE.md files	25
7.2	Create one with /init	26
7.3	What to put in it	26
7.4	Auto memory	27
08	Everyday Commands and Shortcuts	28
8.1	Slash commands	28
8.2	Keyboard shortcuts	29
8.3	Starting Claude Code from the terminal	30
09	Extending Claude Code	31
9.1	MCP: connect to other apps	31
9.2	Skills: your own slash commands	32
9.3	Subagents: specialist helpers	33
9.4	Hooks: automatic actions	33
9.5	Plugins: install bundles	34
10	Beyond the Terminal	35
10.1	The desktop app	35
10.2	VS Code and Cursor	35
10.3	Claude Code on the web and phone	35
10.4	Scripts and automation	36
11	Costs and Limits	37
11.1	How usage works	37
11.2	Make your usage go further	37
12	Troubleshooting	38
	Quick Reference	39
13	The Complete Slash-Command Reference	40
13.1	Three kinds of slash command	41
13.2	/clear: start fresh	41
13.3	/compact: shrink without forgetting	42

13.4	/context: see what's using space	42
13.5	/rewind: the undo button	42
13.6	/resume, /branch, /fork, and /rename	43
13.7	/model and /effort	43
13.8	/init and /memory	43
13.9	/permissions, /sandbox, and /fewer-permission-prompts	44
13.10	Review and quality commands	44
13.11	Housekeeping commands	45
13.12	Recipes: commands that work well together	45
13.13	Every documented command, A–Z	45
14	Command-Line Flags and Shortcuts Reference	52
14.1	The ten flags beginners actually use	53
14.2	Terminal subcommands	53
14.3	Every startup flag	55
14.4	Every keyboard shortcut	58
15	Real-World Workflows, Step by Step	62
15.1	Workflow 1: Fix a bug from an error message	62
15.2	Workflow 2: Add a feature with plan mode	63
15.3	Workflow 3: Learn an unfamiliar project	64
15.4	Workflow 4: Tests first	64
15.5	Workflow 5: Ship a pull request	64
15.6	Workflow 6: Long tasks in the background	65
15.7	Workflow 7: Scripts with <code>claude -p</code>	65
15.8	Daily routine checklist	65
16	Troubleshooting and FAQ	66
16.1	First-aid routine	66
16.2	Symptom table	67
16.3	Frequently asked questions	67
17	Hooks, Settings, and Environment Reference	69
17.1	Settings files and precedence	69
17.2	Hook events	70
17.3	Worked example: format every file Claude edits	71
17.4	Environment variables	71
	Glossary	89

Welcome to Claude Code

Claude Code is an **AI coding agent**. That means it can do more than answer questions: it can open the files in your project, change them, run programs, check its own work, and keep going until a task is done. You stay in charge the whole time.

IN THIS CHAPTER

- What Claude Code is and what it's good for
- The different places you can use it
- What you need before you start

1.1 What is an "agent"?

You have probably used a **chatbot** such as the Claude app or ChatGPT. You type a question and it types back an answer. If it suggests code, *you* copy it, paste it into a file, and run it.

An **agent** closes that loop. When you ask Claude Code to “add a contact form to my website,” it will:

1. Look through your project to understand how it's built.
2. Decide which files need to change.
3. Edit those files (asking your permission first, unless you've told it not to).
4. Run commands, for example starting your website, to check that the change works.
5. Tell you what it did, in plain language.

This cycle of *look* → *plan* → *act* → *check* is called the **agentic loop**. It's what makes Claude Code feel less like a search engine and more like a teammate.

1.2 What can you do with it?

You don't need to be a programmer to get value from Claude Code. People use it to:

- **Build things from a description.** “Make me a one-page website for my bakery with our hours and a menu.”
- **Understand unfamiliar code.** “Explain what this project does, as if I’m new to coding.”
- **Fix problems.** “When I run the app I get this error. Find out why and fix it.”
- **Automate chores.** “Rename all these photos by the date they were taken.”
- **Work with Git and GitHub.** “Commit my changes with a clear message and open a pull request.”
- **Write documents and data files**, not just code: reports, spreadsheets, scripts, and more.

TIP

You can ask Claude Code about itself. Questions like “what can Claude Code do?” or “how do I add an MCP server?” work, because it has its own documentation built in.

1.3 Where Claude Code runs

Claude Code is one product with several “front doors.” They share the same brain and the same settings, so what you learn in this manual works everywhere.

SURFACE	WHAT IT LOOKS LIKE	BEST FOR
Terminal (CLI)	A text window where you type	The full feature set; the focus of this manual
Desktop app	The Claude app’s Code tab	People who prefer buttons and windows over typing commands
VS Code / Cursor extension	A chat panel inside your code editor	Seeing edits side-by-side with your files
JetBrains plugin	Runs inside IntelliJ, PyCharm, etc.	Developers already using those editors
Web	claude.ai/code	Running tasks in the cloud, even from a phone
Slack, GitHub Actions	Mention Claude in a chat or pull request	Teams and automation

CLI stands for *command-line interface*: a program you control by typing text commands instead of clicking.

1.4 What you need

- ☐ A computer running **macOS 13+**, **Windows 10 (1809+)**, or a recent **Linux** (Ubuntu 20.04+, Debian 10+), with at least 4 GB of RAM
- ☐ An internet connection
- ☐ A **paid Claude account**: Pro, Max, Team, or Enterprise, *or* an Anthropic Console account that pays per use
- ☐ A folder to work in (it can be empty, so you don't need an existing project)

NOTE

The free Claude plan does **not** include Claude Code. The cheapest way in is the Pro plan. See the pricing table in Chapter 3.

Terminal Basics in Ten Minutes

The terminal looks intimidating, but you only need a handful of ideas to use Claude Code comfortably. Think of it as a very literal text message conversation with your computer.

IN THIS CHAPTER

- How to open a terminal on Mac, Windows, and Linux
- Folders, paths, and the “current directory”
- The five commands you’ll actually use

2.1 Opening a terminal

OPEN YOUR TERMINAL

1 On a Mac.

Press `Cmd` + `Space`, type `Terminal`, and press `Enter`.

2 On Windows.

Press the `Windows` key, type `PowerShell`, and open **Windows PowerShell** (or **Terminal**, which opens PowerShell by default).

3 On Linux.

Press `Ctrl` + `Alt` + `T` on most systems, or search your apps for “Terminal.”

You’ll see a line ending in something like `%`, `$`, or `>`. That’s the **prompt**: the terminal waiting for you to type. When this manual shows a command, type it after the prompt and press `Enter`.

 TIP

On Windows, the prompt starts with `PS C:\` when you're in **PowerShell** and just `C:\` when you're in the older **Command Prompt (CMD)**. Claude Code works in both, but the install commands differ.

2.2 Folders and paths

Everything on your computer lives in **folders** (also called **directories**). A **path** is the address of a folder or file, like `/Users/sam/projects/bakery` on a Mac or `C:\Users\Sam\projects\bakery` on Windows.

The terminal is always “standing in” one folder, called the **current directory** or **working directory**. Claude Code works on the files in whatever folder you start it from. That's how it knows which project you mean.

The `~` symbol is shorthand for your **home folder** (for example `/Users/sam`).

2.3 The five commands you need

COMMAND	WHAT IT DOES	EXAMPLE
<code>pwd</code>	Show which folder you're in (“print working directory”)	<code>pwd</code>
<code>ls</code>	List the files here (<code>dir</code> in CMD)	<code>ls</code>
<code>cd</code>	Change into a folder	<code>cd projects/bakery</code>
<code>cd ..</code>	Go up one folder	<code>cd ..</code>
<code>mkdir</code>	Make a new folder	<code>mkdir bakery</code>

Here's how to make a practice folder and step inside it:

TERMINAL

```
cd ~
mkdir my-first-project
cd my-first-project
pwd
```

 TIP

Press `Tab` while typing a folder name and the terminal finishes it for you. Press the `↑` arrow to bring back the last command you typed.

2.4 Copying and pasting in a terminal

- **Mac:** `Cmd` + `C` and `Cmd` + `V`, as usual.
- **Windows Terminal / PowerShell:** `Ctrl` + `V` pastes; right-click also pastes.
- **Linux:** usually `Ctrl` + `Shift` + `V`.

In a terminal, `Ctrl` + `C` usually means **stop the running program**, not copy.

Install and Sign In

Installation takes about two minutes. Anthropic’s recommended method is the **native installer**, which also keeps Claude Code up to date automatically.

IN THIS CHAPTER

- Installing on macOS, Linux, and Windows
- Choosing a plan and logging in
- Checking that everything works

3.1 Install with the native installer

INSTALL CLAUDE CODE

- 1 Open a terminal.**
See Chapter 2 if you’re not sure how.
- 2 Paste the command for your system and press Enter.**
macOS, Linux, or WSL:

TERMINAL

```
curl -fsSL https://claude.ai/install.sh | bash
```

Windows PowerShell:

POWERSHELL

```
irm https://claude.ai/install.ps1 | iex
```

Windows Command Prompt (CMD):

COMMAND PROMPT

```
curl -fsSL https://claude.ai/install.cmd -o install.cmd && install.cmd && del install.cmd
```

3

Wait.

The installer shows no progress bar while it downloads. That's normal.

4

Open a *new* terminal window

so it picks up the new program.

5

Check the install.

Type `claude --version`. You should see a version number followed by (Claude Code).

⚠️ "COMMAND NOT FOUND"?

If your terminal says `claude` isn't found or isn't recognized, the install folder isn't on your **PATH** yet. (The PATH is the list of folders your terminal searches for programs.) Close every terminal window and open a fresh one first. If it still fails, run the installer again and read its last few lines. It prints the exact fix.

Other ways to install

METHOD	COMMAND	UPDATES
Homebrew (Mac)	<code>brew install --cask claude-code</code>	Manual: <code>brew upgrade claude-code</code>
WinGet (Windows)	<code>winget install Anthropic.ClaudeCode</code>	Manual: <code>winget upgrade Anthropic.ClaudeCode</code>
npm (needs Node.js 22+)	<code>npm install -g @anthropic-ai/claude-code</code>	Manual
apt / dnf / apk (Linux)	See the setup page in the official docs	Through your system updates

⚠️ DANGER

Never install with `sudo npm install -g`. Running npm as an administrator causes permission problems and is a security risk.

Windows notes

- You do **not** need to run the installer as Administrator.
- Installing **Git for Windows** (from git-scm.com) is recommended. It gives Claude Code a Bash shell to run commands with. Without it, Claude Code uses PowerShell instead, which also works.
- If you already use **WSL** (Windows Subsystem for Linux), install from inside your WSL terminal using the macOS/Linux command.

3.2 Choose a plan

Claude Code needs a paid account. Prices below are from claude.com/pricing in October 2026:

PLAN	PRICE	WHO IT'S FOR
Free	\$0	Chat only, no Claude Code
Pro	\$20/month (\$17/month billed yearly)	Learning and light daily use
Max 5x	\$100/month	Heavy daily use: 5× Pro's usage
Max 20x	\$200/month	All-day use: 20× Pro's usage
Team	\$25/seat/month Standard, \$125 Premium (less yearly)	Small teams, 2–150 people
Enterprise	Custom	Large organizations
Console (API)	Pay per use	Developers who prefer metered billing

TIP

Start with **Pro**. If you regularly hit your usage limit, upgrade to Max later. Inside Claude Code, `/usage` shows how much of your limit you've used.

3.3 Log in for the first time

SIGN IN

- 1 Go to your project folder.**
For example: `cd ~/my-first-project`.
- 2 Start Claude Code.**
Type `claude` and press `Enter`.
- 3 Pick a theme.**
Use the arrow keys to choose light or dark.

- 4 **Choose how to log in.**
Pick your Claude subscription (Pro/Max/Team/Enterprise) or Anthropic Console.
- 5 **Approve in your browser.**
A browser window opens; click **Authorize**. Return to the terminal.
- 6 **Trust the folder.**
Claude Code asks whether you trust the files in this folder. For your own projects, choose **Yes**.

You only log in once; Claude Code remembers your credentials. To switch accounts later, type `/login` inside Claude Code.

3.4 Health check and updates

- `claude doctor` checks your installation and settings and suggests fixes. Run it whenever something feels off.
- Native installs **update themselves** in the background. To update right now: `claude update`.
- Prefer slower, battle-tested releases? Type `/config` and set **Auto-update channel** to **stable** (about a week behind, skipping releases with major bugs).

Your First Session

Let's build something. In this chapter you'll start Claude Code, ask it questions, and have it create and change real files.

IN THIS CHAPTER

- What the Claude Code screen shows you
- Asking questions and requesting changes
- Approving actions, interrupting, and undoing

4.1 The Claude Code screen

When you run `claude`, you'll see:

- A **welcome panel** showing the version, the AI model in use, and your current folder.
- A **prompt box** where you type, marked with `>`.
- A **status bar** underneath showing the current *permission mode* (more on this in Chapter 6).

Type your request in plain English and press `Enter`. There's no special syntax to learn.

TIP

To write several lines before sending, press `Shift` + `Enter` for a new line (or type `\` then `Enter`, which works in every terminal).

4.2 Build a small website

YOUR FIRST PROJECT

1 Start in an empty folder.

TERMINAL

```
mkdir ~/bakery-site  
cd ~/bakery-site  
claude
```

2 Describe what you want.

PROMPT

Create a simple one-page website for a bakery called "Rise & Shine". Include our hours (Tue–Sun, 7am–3pm), a short menu with prices, and a contact section. Keep it to plain HTML and CSS.

3 Watch it plan and work.

Claude explains what it's about to do, then creates files such as `index.html` and `styles.css`.

4 Approve when asked.

If Claude asks permission to create a file, choose **Yes**. (Chapter 6 explains how to approve whole categories at once.)

5 Open the result.

Ask: `open the website in my browser`. Or open `index.html` from your file manager.

6 Change it.

Try: `make the colors warmer and add a photo placeholder at the top`.

That's the whole rhythm: **describe, review, refine**.

4.3 Ask questions about existing code

Claude Code shines when you point it at a project someone else wrote. Start it inside the project folder and try:

PROMPT

what does this project do?

PROMPT

explain the folder structure like I'm new to coding

PROMPT

where is the main entry point, and what happens when the app starts?

You don't need to tell Claude which files to read. It looks around by itself. If you *want* to point at a file, type `@` and start typing its name; Claude Code autocompletes the path.

4.4 Approving, interrupting, and undoing

Permission prompts. Before Claude edits a file or runs a command it isn't sure about, it may show what it intends to do and offer choices like:

- **Yes:** allow this one action.
- **Yes, and don't ask again:** allow this kind of action from now on in this project.
- **No:** refuse, and optionally tell Claude what to do instead.

Interrupting. Press `Esc` at any time to stop Claude mid-task. It keeps what it has done so far, and you can redirect it: "Stop. Use blue instead of green."

Undoing. Claude Code saves a **checkpoint** before each change. Press `Esc` twice on an empty prompt (or type `/rewind`) to roll the conversation *and/or* your files back to an earlier point.

⚠ WARNING

Checkpoints cover file edits Claude makes. They can't undo things outside your computer, like a sent email, a deleted cloud file, or a database change. For real safety, use Git (Chapter 6 and the GitHub manual in this series).

4.5 Ending and resuming

TO...	DO THIS
Leave Claude Code	Type <code>/exit</code> , or press <code>Ctrl</code> + <code>D</code> twice
Start fresh in the same window	<code>/clear</code>
Continue your last conversation	<code>claude -c</code> (from the terminal)
Pick an older conversation	<code>claude -r</code> , or <code>/resume</code> inside Claude Code
Name a session to find it later	<code>/rename bakery-site</code>

Talking to Claude Effectively

Claude Code is capable, but it isn't a mind reader. A little care in how you ask makes a huge difference in what you get back.

IN THIS CHAPTER

- The habits that produce great results
- Plan mode: think first, then build
- Sharing screenshots, errors, and files

5.1 Five habits of great prompts

1. **Be specific.** Instead of “fix the bug,” say “users see a blank screen after entering a wrong password, so fix that.”
2. **Give the goal, not just the task.** “I want customers to order ahead online” helps Claude make better choices than “add a form.”
3. **Break big jobs into steps.** Number them, and Claude will work through them in order.
4. **Let Claude explore first.** “Read the code and explain how login works before changing anything.”
5. **Say how you'll judge success.** “When you're done, run the tests and show me they pass.”

PROMPT

1. Add a "Order ahead" page with a form for name, item, and pickup time.
2. Save orders to a file called orders.csv.
3. Add a link to the new page in the site's menu.
4. Open the site and confirm the form works before you finish.

5.2 Plan mode: look before you leap

For anything bigger than a small tweak, ask Claude to **plan first**. In **plan mode** Claude researches and writes a step-by-step plan but won't edit your files until you approve it.

- Press `Shift` + `Tab` until the status bar says `plan mode on`, **or**
- Start your message with `/plan`, for example: `/plan add online ordering`.

When the plan appears you can approve it, ask for changes (“No, keep planning, and don't use a database”), or press `Ctrl` + `G` to edit the plan yourself in a text editor.

TIP

Plan mode is the single best habit for beginners. Reading the plan teaches you how the project works, and it catches misunderstandings before any files change.

5.3 Share what you see

- **Error messages:** copy and paste them straight into the prompt. Include the whole thing.
- **Screenshots:** paste an image with `Ctrl` + `V` (`Cmd` + `V` in iTerm2, `Alt` + `V` on Windows). Claude can see it.
- **Files:** type `@` followed by part of a file name.
- **A command's output:** start a line with `!` to run a shell command yourself, for example `!npm test`. Claude sees the result and responds to it.

5.4 Ask a side question

Want to ask something without derailing the current task? Use `/btw`:

PROMPT

```
/btw what's the difference between HTML and CSS?
```

The answer appears without being added to the main conversation.

5.5 Keep conversations focused

Claude can hold a lot in mind at once, but that space, the **context window**, is not unlimited. Long, wandering sessions get slower and cost more of your usage.

- Use `/clear` when you switch to an unrelated task.
- Use `/compact` to summarize a long conversation and free up space while staying on the same task.
- Use `/context` to see how full the context window is.

Permissions and Safety

Claude Code can edit files and run commands on your computer, so it comes with a permission system: rules for what it may do without asking. Understanding it lets you work faster *and* safer.

IN THIS CHAPTER

- The six permission modes and how to switch
- Allow, ask, and deny rules
- Safety nets: checkpoints, Git, and the sandbox

6.1 Permission modes

A **permission mode** sets how much Claude can do before checking with you. Press `Shift` + `Tab` to cycle modes during a session. The status bar shows the active one.

MODE	RUNS WITHOUT ASKING	GOOD FOR
Manual (<code>default</code>)	Reading files only	Sensitive work; learning what Claude does
Accept edits (<code>acceptEdits</code>)	Reading, editing files, and simple file commands like <code>mkdir</code> , <code>mv</code> , <code>cp</code>	Iterating on code you'll review afterwards
Plan (<code>plan</code>)	Reading and exploring; no edits until you approve a plan	Thinking before building
Auto (<code>auto</code>)	Almost everything, with a safety classifier checking each action in the background	Long tasks with few interruptions
Don't ask (<code>dontAsk</code>)	Only pre-approved tools; everything else is refused	Scripts and automation
Bypass (<code>bypassPermissions</code>)	Everything, no checks	Disposable containers and virtual machines only

Auto mode is the starting mode for new terminal and VS Code sessions on recent versions of Claude Code. A second AI model (the **classifier**) reviews each risky action, such as running a script from the internet, force-pushing code, deploying, or deleting files that existed before the session, and blocks anything that goes beyond what you asked for.

☆ **IMPORTANT**

You can tell auto mode about limits in plain English: “don’t push anything until I’ve reviewed it.” The classifier treats that as a rule for the rest of the conversation. For a hard guarantee, add a **deny rule** (below).

To start a session in a particular mode:

TERMINAL

```
claude --permission-mode plan
```

6.2 Allow, ask, and deny rules

Rules let you pre-approve safe actions and forbid dangerous ones. Type `/permissions` to see and edit them, or write them in a settings file.

- **Allow:** run without asking.
- **Ask:** always ask first, even in auto mode.
- **Deny:** never allowed, in any mode.

Deny rules win over ask rules, which win over allow rules.

.CLAUDE/SETTINGS.JSON

```
{
  "permissions": {
    "allow": [
      "Bash(npm run *)",
      "Bash(git commit *)"
    ],
    "ask": [
      "Bash(git push *)"
    ],
    "deny": [
      "Read(./env)",
      "Bash(rm -rf *)"
    ]
  }
}
```

The `*` matches anything, so `Bash(npm run *)` covers `npm run build`, `npm run test`, and so on. Put the `*` *after* the sub-command: `Bash(git commit *)` is narrow, but `Bash(git *)` allows every Git command.

 TIP

Tired of approving the same harmless commands? Type `/fewer-permission-prompts`. Claude scans your recent sessions and proposes a safe allow-list for read-only commands you use often.

6.3 Where settings live

Claude Code reads settings from several files. When the same setting appears in more than one place, the higher one in this list wins:

PRIORITY	FILE	APPLIES TO
1 (highest)	Managed settings from your organization	Everyone in your company
2	<code>claude --settings ...</code> on the command line	This session
3	<code>.claude/settings.local.json</code>	You, in this project (not shared)
4	<code>.claude/settings.json</code>	Everyone working in this project
5	<code>~/.claude/settings.json</code>	You, in every project

`~` means your home folder. A file inside `.claude/` lives in the project folder.

6.4 Safety nets

- **Checkpoints:** `/rewind` (or `Esc Esc`) restores earlier file versions.
- **Git:** commit your work before big changes so you can always get back. Ask Claude: “commit everything with a clear message before you start.”
- **Protected files:** Claude never silently edits things like `.git`, `.bashrc`, or its own `.claude` settings. It always asks, except in bypass mode.
- **Sandbox:** type `/sandbox` (macOS, Linux, WSL2) to run Claude’s shell commands in an isolated box with limited file and network access.

 BYPASS MODE IS FOR THROWAWAY MACHINES

`--dangerously-skip-permissions` turns off every check. A bad instruction hidden in a web page or file (a **prompt injection**) could then make Claude do real damage. Only use it inside a container or virtual machine you’re happy to wipe. The *Maximum Autonomy* guide in this series covers how to set that up.

Memory: Teaching Claude Your Project

Every session starts with a blank slate. Two features carry knowledge from one session to the next: **CLAUDE.md files**, which you write, and **auto memory**, which Claude writes.

IN THIS CHAPTER

- What a CLAUDE.md file is and where to put it
- Generating one with `/init`
- What belongs in it, with an example

7.1 CLAUDE.md files

A `CLAUDE.md` file is a plain-text note, written in **Markdown** (a simple way to format text with `#` headings and `-` bullets), that Claude reads at the start of every session. Use it for anything you'd otherwise re-explain each time: how to run the project, house style, "always do X" rules.

WHERE	FILE	WHO IT'S FOR
Your whole computer	<code>~/.claude/CLAUDE.md</code>	Your personal preferences, every project
A project (shared)	<code>./CLAUDE.md</code> or <code>~/.claude/CLAUDE.md</code>	Everyone on the project, checked into Git
A project (just you)	<code>./CLAUDE.local.md</code>	Your private notes; add it to <code>.gitignore</code>

7.2 Create one with /init

Inside a project, type:

PROMPT

```
/init
```

Claude studies the project and writes a starter `CLAUDE.md` with build commands, test instructions, and conventions it finds. If one already exists, it suggests improvements instead of overwriting.

7.3 What to put in it

CLAUDE.MD

```
# Rise & Shine bakery website

## How to preview
- Open index.html in a browser. No build step.

## Style
- Plain HTML and CSS only. No frameworks.
- Colors: warm cream (#FFF6E9) and cinnamon (#B5562B).
- Write friendly, short copy. No jargon.

## Rules
- Keep every page working on phones.
- Ask before deleting any image.
```

Tips from Anthropic's guidance:

- **Keep it short:** aim for under about 200 lines. Long files get followed less reliably.
- **Be concrete:** “Use 2-space indentation” beats “format code nicely.”
- **Add a line when Claude makes the same mistake twice.**
- **Import other files** with `@`, e.g. `See @README.md for an overview.`

NOTE

CLAUDE.md is *guidance*, not a lock. To truly block an action, use a deny rule (Chapter 6) or a hook (Chapter 9).

7.4 Auto memory

As you work, Claude can note your corrections and preferences (“Sam prefers short commit messages”) in its own memory files, which load automatically next time. Type `/memory` to view or edit them, open your CLAUDE.md files, or turn auto memory on or off.

CHAPTER 8

Everyday Commands and Shortcuts

You don't need to memorize these. Type `/` at any time to see a searchable menu of everything available. Here are the ones you'll use most.

8.1 Slash commands

COMMAND	WHAT IT DOES
<code>/help</code>	Show help and available commands
<code>/init</code>	Create a <code>CLAUDE.md</code> for this project
<code>/clear</code>	Start a new, empty conversation
<code>/compact</code>	Summarize the conversation to free up space
<code>/context</code>	Show how full the context window is
<code>/model</code>	Switch AI model (e.g. <code>opus</code> , <code>sonnet</code> , <code>haiku</code>)
<code>/plan</code>	Enter plan mode, optionally with a task
<code>/permissions</code>	View and edit allow / ask / deny rules
<code>/rewind</code>	Roll back the conversation and/or files
<code>/resume</code>	Reopen an earlier conversation
<code>/diff</code>	Review all changes made so far
<code>/review</code>	Ask Claude to review your changes for bugs
<code>/memory</code>	Edit CLAUDE.md files and auto memory
<code>/mcp</code>	Manage connected MCP servers (Chapter 9)
<code>/usage</code>	See plan limits and usage (<code>/cost</code> works too)

<code>/config</code>	Open settings: theme, model, editor mode...
<code>/doctor</code>	Diagnose and fix setup problems
<code>/add-dir</code>	Let Claude work in an extra folder
<code>/export</code>	Save the conversation as text
<code>/btw</code>	Ask a side question
<code>/exit</code>	Quit

8.2 Keyboard shortcuts

KEYS	ACTION
<code>Esc</code>	Stop Claude mid-task
<code>Esc</code> <code>Esc</code>	Clear your draft; on an empty prompt, open rewind
<code>Shift</code> + <code>Tab</code>	Cycle permission modes
<code>Shift</code> + <code>Enter</code>	New line without sending
<code>↑</code> / <code>↓</code>	Scroll through previous prompts
<code>Ctrl</code> + <code>R</code>	Search prompt history
<code>Ctrl</code> + <code>C</code>	Interrupt; on an empty prompt, press twice to exit
<code>Ctrl</code> + <code>O</code>	Show the detailed transcript of what Claude did
<code>Ctrl</code> + <code>G</code>	Edit your prompt (or a plan) in a text editor
<code>Ctrl</code> + <code>V</code>	Paste an image
<code>Ctrl</code> + <code>B</code>	Send a long-running command to the background
<code>?</code>	Show the shortcut help panel (on an empty prompt)

8.3 Starting Claude Code from the terminal

COMMAND	WHAT IT DOES
<code>claude</code>	Start an interactive session
<code>claude "fix the build error"</code>	Start with a first request
<code>claude -c</code>	Continue the most recent conversation here
<code>claude -r</code>	Choose an older conversation to resume
<code>claude -p "summarize README.md"</code>	Answer once and exit (handy in scripts)
<code>claude --model opus</code>	Use a specific model this session
<code>claude --permission-mode plan</code>	Start in a specific permission mode
<code>claude -w new-feature</code>	Work in an isolated copy of the repo (a Git <i>worktree</i>)
<code>claude update</code>	Update now
<code>claude doctor</code>	Check installation health

Extending Claude Code

Out of the box, Claude Code can work with files, run commands, and search the web. You can extend it in five ways. Each has its own job:

FEATURE	IN ONE SENTENCE	LIVES IN
MCP servers	Connect Claude to outside tools: GitHub, Notion, databases, browsers	<code>claude mcp add ...</code>
Skills	Save a reusable set of instructions as a <code>/command</code>	<code>.claude/skills/<name>/SKILL.md</code>
Subagents	Specialist helpers with their own instructions and tools	<code>.claude/agents/<name>.md</code>
Hooks	Run your own command automatically when something happens	<code>"hooks"</code> in <code>settings.json</code>
Plugins	Bundles of the above you can install in one step	<code>/plugin</code>

9.1 MCP: connect to other apps

MCP (Model Context Protocol) is an open standard that lets AI tools plug into other software. An **MCP server** is a small connector that gives Claude new abilities, such as reading your Notion pages or creating GitHub issues. (This series has a whole manual on MCP.)

Most modern services offer a **remote** server: a web address. Add one like this:

TERMINAL

```
claude mcp add --transport http notion https://mcp.notion.com/mcp
```

Then type `/mcp` inside Claude Code, select the server, and sign in through your browser.

Some servers run as a program on your own computer (a **local** or **stdio** server). Everything after the `--` is the command that starts it:

TERMINAL

```
cclaude mcp add --env API_KEY=your-key myserver -- npx -y @example/mcp-server
```

COMMAND	WHAT IT DOES
<code>cclaude mcp list</code>	List servers and whether they're connected
<code>cclaude mcp get <name></code>	Show details for one server
<code>cclaude mcp remove <name></code>	Remove a server
<code>--scope user</code>	Make a server available in all your projects
<code>--scope project</code>	Save it in <code>.mcp.json</code> to share with your team

 WARNING

Only install MCP servers you trust. A server can see what Claude sends it and can return text that tries to trick Claude. Prefer official servers published by the company whose product you're connecting.

9.2 Skills: your own slash commands

A **skill** is a folder containing a `SKILL.md` file of instructions. Claude uses it automatically when it fits the situation, or you run it by name.

```
~/CLAUDE/SKILLS/WEEKLY-UPDATE/SKILL.MD
```

```
---
```

```
description: Writes a friendly weekly progress update from recent Git commits. Use when asked for a  
status update or weekly summary.
```

```
---
```

```
Look at the Git commits from the last 7 days. Write a short update with:
```

- three bullet points of what got done
- one bullet of what's next

```
Use plain language a non-programmer would understand.
```

Now `/weekly-update` works in every project. Put the folder in a project's `.claude/skills/` instead to share it with your team.

9.3 Subagents: specialist helpers

A **subagent** is a helper Claude can hand work to. Each has its own instructions, its own allowed tools, and its own fresh context, so a big research job doesn't clutter your main conversation. The easiest way to make one is to ask:

PROMPT

Create a project subagent called "copy-editor" that checks my website text for spelling, tone, and reading level. It should only be able to read files.

Claude saves it as `.claude/agents/copy-editor.md`. Claude Code also ships with built-in subagents, such as **Explore** (fast, read-only searching) and **Plan**.

9.4 Hooks: automatic actions

A **hook** runs a command of yours at a specific moment, for example after every file edit or when Claude needs your attention. Unlike `CLAUDE.md`, hooks *always* run. This example shows a desktop notification on a Mac whenever Claude is waiting for you:

~/CLAUDE/SETTINGS.JSON

```
{
  "hooks": {
    "Notification": [
      {
        "matcher": "",
        "hooks": [
          {
            "type": "command",
            "command": "osascript -e 'display notification \"Claude needs you\" with title \"Claude Code\"'"
          }
        ]
      }
    ]
  }
}
```

Type `/hooks` to see what's configured. You can also just describe the hook you want and ask Claude to set it up.

9.5 Plugins: install bundles

A **plugin** packages skills, subagents, hooks, and MCP servers together. Type `/plugin` to browse and install from **marketplaces** (catalogs of plugins). Anthropic's official marketplace is built in.

Beyond the Terminal

Everything you've learned carries over to Claude Code's other surfaces.

10.1 The desktop app

Download the Claude app from claude.com/download, sign in, and open the **Code** tab. You get the same Claude Code with a graphical interface: a folder picker, a visual **diff** (a side-by-side view of what changed), a permission-mode selector next to the send button, and the ability to run several sessions in parallel.

10.2 VS Code and Cursor

In VS Code or Cursor, open the Extensions view (`Ctrl`/`Cmd` + `Shift` + `X`), search for **Claude Code**, and click **Install**. A Claude panel appears in the sidebar. You can review proposed changes as inline diffs and @-mention the lines you've selected.

10.3 Claude Code on the web and phone

At claude.ai/code, or in the **Code** tab of the Claude mobile app, you can start a **cloud session**: Claude works on a copy of your GitHub repository on Anthropic's computers, so it keeps going even when your laptop is closed. When it's done, it pushes a branch you can review. Cloud sessions are available on Pro, Max, Team, and eligible Enterprise seats, and need your GitHub account connected.

From the terminal, `claude --cloud "fix the login bug"` hands a task to the cloud, and `claude --teleport` pulls a cloud session back to your computer.

10.4 Scripts and automation

`claude -p` (“print mode”) answers once and exits, which makes Claude Code usable inside scripts:

TERMINAL

```
cat error.log | claude -p "explain this error in one paragraph"
```

For GitHub, type `/install-github-app` to let your team mention `@claude` in issues and pull requests to get work done automatically.

Costs and Limits

11.1 How usage works

On a **subscription** (Pro, Max, Team), Claude Code use is included up to your plan's limits, which reset over time. On a **Console** account you pay per **token** (a token is a chunk of text, roughly $\frac{3}{4}$ of a word). Longer conversations and bigger files use more tokens.

Type `/usage` to see your current usage, your limits, and a breakdown of what's using the most.

11.2 Make your usage go further

- `/clear` **between unrelated tasks**. Old conversation history is re-read with every message.
- `/compact` **during long tasks** to shrink the history.
- **Use a lighter model for simple jobs:** `/model haiku` or `/model sonnet` for routine work, and save the most powerful model for hard problems.
- **Be specific.** A precise request avoids wasted exploration.
- **Keep CLAUDE.md lean.** It's loaded into every session.

TIP

If you hit a limit mid-task, `/rate-limit-options` shows your choices: wait for the reset, add usage credits, or upgrade.

Troubleshooting

PROBLEM	TRY THIS
<code>claude: command not found</code>	Open a new terminal window. Still failing? Re-run the installer and follow the PATH fix it prints, or run <code>claude doctor</code> .
<code>'irm' is not recognized</code>	You're in CMD, not PowerShell. Use the CMD install command, or open PowerShell.
The token '&&' is not a valid statement separator	You're in PowerShell. Use the PowerShell command instead.
Login window doesn't open	Copy the link Claude Code prints into your browser manually.
Claude keeps asking permission for the same thing	Choose Yes, and don't ask again , add an allow rule, or run <code>/fewer-permission-prompts</code> .
Claude ignores an instruction	Put it in CLAUDE.md, make it concrete, and keep the file short. For must-never-happen rules, use a deny rule.
Responses getting slow or confused	<code>/compact</code> or <code>/clear</code> , then restate the goal.
Something is broken and you don't know why	<code>/doctor</code> inside Claude Code. It diagnoses and offers fixes.
MCP server shows "Failed to connect"	<code>claude mcp get <name></code> shows the error; re-authenticate in <code>/mcp</code> .

NOTE

Stuck? Type `/bug` to report a problem to Anthropic, ask Claude itself ("why did that fail?"), or take the free Claude Code courses at academy.claude.com.

Quick Reference

Install (Mac/Linux): `curl -fsSL https://claude.ai/install.sh | bash`

Install (Windows PowerShell): `irm https://claude.ai/install.ps1 | iex`

Start: `cd my-project` then `claude` · **Continue:** `claude -c` · **Resume:** `claude -r`

Stop: `Esc` · **Undo:** `Esc` `Esc` or `/rewind` · **Modes:** `Shift` + `Tab` · **New line:** `Shift` + `Enter`

Plan first: `/plan ...` · **Fresh start:** `/clear` · **Shrink history:** `/compact` · **Project memory:** `/init`

Connect an app: `claude mcp add --transport http <name> <url>` then `/mcp` to sign in

Health check: `claude doctor` · **Update:** `claude update` · **Usage:** `/usage`

The Complete Slash-Command Reference

Slash commands are the control panel of Claude Code. You type `/` at the start of a message and a searchable menu appears. This chapter lists **every built-in command and bundled skill documented by Anthropic** as of this edition, so you can look any of them up on paper.

IN THIS CHAPTER

- How slash commands, bundled skills, and your own skills differ
- Deep dives on the commands you'll use every day: `/clear`, `/compact`, `/context`, `/rewind`, `/resume`, `/model`, `/init`, `/memory`, `/permissions`
- A full A-Z table of every documented command
- Recipes that chain commands together

WHERE THIS LIST COMES FROM

The A-Z table was generated from Anthropic's official commands page, code.claude.com/docs/en/commands, in October 2026. Claude Code updates often. Some commands appear only on certain plans, platforms, or versions. If a command here doesn't show up for you, run `claude update`, then type `/` and search for it. Your own `/` menu is always the final word.

13.1 Three kinds of slash command

KIND	WHAT IT IS	EXAMPLE
Built-in command	Fixed logic inside Claude Code. It runs instantly and usually doesn't cost tokens.	<code>/clear</code> , <code>/model</code> , <code>/exit</code>
Bundled skill	A ready-made set of instructions that Claude follows using its tools. It uses tokens, like a normal request.	<code>/code-review</code> , <code>/debug</code> , <code>/batch</code>
Your own skill	A <code>SKILL.md</code> file you or your team wrote (Chapter 9).	<code>/deploy</code> , <code>/fix-issue</code> 123

Anything you type after the command name is passed in as **arguments**. For example, in `/compact keep the API decisions`, the text after `/compact` tells the summary what to focus on.

TIP

A command only runs when it's at the **start** of your message. If you write "please `/deploy` this" in the middle of a sentence, nothing runs on its own. Claude just treats it as permission to use that skill if it decides to.

13.2 `/clear`: start fresh

`/clear` wipes the conversation and starts a new one with empty context. Your files are **not** touched. Only Claude's short-term memory of the chat is reset. `CLAUDE.md` loads again automatically.

USE `/CLEAR` BETWEEN TASKS

- 1 Finish or commit the current task.**
Ask Claude to summarize what it did if you want a record.
- 2 Type `/clear`.**
Optionally add a name, such as `/clear login-work`, to label the old conversation so you can find it later.
- 3 Start the next task.**
Claude now has a clean, fast, cheap context.
- 4 Need the old chat back?**
Use `/resume` and pick it from the list.

Use `/clear` when you switch to an unrelated task, when Claude keeps repeating an old mistake, or when the conversation has drifted. `/reset` and `/new` are aliases.

13.3 /compact: shrink without forgetting

Every message, file, and command output takes up room in the **context window**, the AI's working memory. When it fills up, answers get slower, more expensive, and less sharp.

`/compact` replaces the conversation so far with a summary, so you keep the important facts and free most of the space.

PROMPT

`/compact` Keep the database schema decisions and the list of failing tests. Drop the exploration of old files.

SITUATION	USE
Same task, but the chat is long	<code>/compact</code>
New, unrelated task	<code>/clear</code>
Claude went down a wrong path a few steps ago	<code>/rewind</code>
You want a side answer without cluttering the chat	<code>/btw</code>

Claude Code also **auto-compacts** when the window is nearly full. `/autocompact` lets you set how full it gets before that happens. Pass a size such as `500k`, or `auto` to go back to the default.

⚠ SUMMARIES LOSE DETAIL

Compaction is lossy. Exact error messages, file paths, and small decisions can disappear. Before compacting, put anything that must survive into `CLAUDE.md` or a notes file, or list it in your `/compact` instructions.

13.4 /context: see what's using space

`/context` draws a coloured grid of your context window, broken down by system prompt, tools, MCP servers, memory files, skills, and messages. It also suggests what to trim. Check it whenever Claude feels slow or forgetful. If MCP tools take up a big block, disconnect the servers you aren't using with `/mcp`.

13.5 /rewind: the undo button

`/rewind` (aliases `/checkpoint` and `/undo`) rolls the conversation, your code, or both back to an earlier message. Pressing `Esc` twice on an empty prompt opens the same menu.

UNDO A BAD CHANGE

- 1 **Type** `/rewind`.
A list of your earlier messages appears.
- 2 **Pick the last good point.**
Choose the message *before* things went wrong.
- 3 **Choose what to restore.**
Code only, conversation only, or both.
- 4 **Re-ask, more precisely.**
Add the detail that was missing the first time.

⚠ DANGER

Checkpoints track edits Claude made with its file tools. Changes made by shell commands, such as `rm` or a database migration, and edits from background forked skills are **not** undone. Git is your real safety net, so commit often.

13.6 /resume, /branch, /fork, and /rename

COMMAND	USE IT TO...
<code>/resume</code>	Reopen an earlier conversation by name or from a picker
<code>/rename my-feature</code>	Give this session a memorable name
<code>/branch</code>	Split the conversation here so you can try another direction, keeping the original
<code>/fork</code>	Copy the conversation into a new background session and keep working here

13.7 /model and /effort

`/model` switches which Claude model you're using and saves it as the default for new sessions. `/effort` sets how hard the model thinks: `low`, `medium`, `high`, `xhigh`, `max`, or `auto`. Higher effort is slower and uses more of your plan, but it's better for tricky bugs and design work. A good habit is to plan with a strong model at high effort, then do routine edits at a lower setting.

13.8 /init and /memory

`/init` reads your project and writes a starter `CLAUDE.md`. `/memory` opens your `CLAUDE.md` files for editing and lets you turn **auto memory** on or off and browse what it has saved. Run `/init` once per project, then keep the file current with `/memory`.

13.9 /permissions, /sandbox, and /fewer-permission-prompts

`/permissions` shows and edits your allow, ask, and deny rules (Chapter 6). `/sandbox` toggles sandbox mode on supported systems. The bundled `/fewer-permission-prompts` skill scans your history for safe, read-only commands you approve often, then suggests allow rules for them. Review that list before you accept it.

13.10 Review and quality commands

COMMAND	WHAT IT'S FOR
<code>/diff</code>	See every change in your working tree
<code>/code-review</code> (alias <code>/review</code>)	Have Claude review your diff, a branch, or a PR for bugs
<code>/security-review</code>	Scan your branch's changes for security risks
<code>/simplify</code>	Run parallel reviewers that tidy up changed code
<code>/verify</code> and <code>/run</code>	Build and run your app to confirm a change actually works

13.11 Housekeeping commands

`/doctor` runs a setup checkup and offers fixes. `/usage` (aliases `/cost` and `/stats`) shows plan limits and spending. `/status` shows your version, model, and account. `/config` opens settings. `/export` saves the chat as text, `/copy` copies the last answer, and `/release-notes` shows what's new. `/exit` (or `/quit`) leaves.

13.12 Recipes: commands that work well together

Long feature, keep it sharp: `/plan` → approve → work → `/compact` keep decisions → continue → `/code-review` → commit.

Switching tasks: commit → `/clear new-task` → `/context` to confirm it's empty.

Undo a wrong turn: `/rewind` → restore code and conversation → re-prompt with more detail.

Before a pull request: `/diff` → `/code-review` → `/security-review` → `/verify`.

Something's broken in Claude Code itself: `/doctor` → `/debug` describe the problem → `/bug` if it's a real bug.

Running low on plan usage: `/usage` → `/model` to a lighter model → `/effort low` for routine edits.

13.13 Every documented command, A–Z

The **Kind** column shows whether the command is built in, a bundled skill (uses tokens as Claude works), or a workflow. Square brackets `[ ]` mean the argument is optional. Angle brackets `< >` mean it's required.

COMMAND	KIND	WHAT IT DOES
<code>/add-dir <path></code>	Built-in	Add a working directory for file access during the current session.
<code>/advisor [model off]</code>	Built-in	Enable or disable the advisor tool, which consults a second model for guidance at key moments during a task.
<code>/agents</code>	Built-in	Print a reminder to ask Claude to create or manage subagents, or to edit <code>.claude/agents/</code> or <code>~/.claude/agents/</code> directly.
<code>/artifact-capabilities</code>	Skill	Load the reference for the runtime capabilities a published artifact can use, such as calling your connectors or offering a file download, including which ones your account has.
<code>/artifact-diagramming</code>	Skill	Load diagramming guidance for Claude to follow in artifacts: when a diagram helps, what to draw, and how to write inline SVG that stays legible in light and dark themes.

<code>/artifacts</code>	Built-in	List the artifacts you own or that are shared with you, then attach one to the session, open it in your browser, or copy its link.
<code>/auto-mode-setup</code>	Built-in	Draft <code>autoMode.environment</code> entries from your project and recent sessions, then review the draft and save it to your user settings.
<code>/autocompact [auto <tokens>]</code>	Built-in	Set the auto-compact window: how full the context window gets before Claude Code compacts automatically.
<code>/autofix-pr [prompt]</code>	Built-in	Spawn a cloud session that watches the current branch's PR and pushes fixes when CI fails or reviewers leave comments.
<code>/background [prompt]</code>	Built-in	Detach the current session to run as a background agent and free this terminal. Alias: <code>/bg</code> .
<code>/batch <instruction></code>	Skill	Orchestrate large-scale changes across a codebase in parallel.
<code>/branch [name]</code>	Built-in	Create a branch of the current conversation at this point, so you can try a different direction without losing the conversation as it stands.
<code>/btw [question]</code>	Built-in	Ask a side question about the current session without adding to the conversation.
<code>/bug [report]</code>	Built-in	Report a bug or share your conversation. Alias: <code>/share</code> .
<code>/cd <path></code>	Built-in	Move this session to a new working directory, keeping the conversation.
<code>/chrome</code>	Built-in	Configure Claude in Chrome settings.
<code>/claude-api [migrate upgrade managed-agents-onboard prompt-audit cost-optimize build-eval hillclimb preserved-thinking-migration]</code>	Skill	Load Claude API and Managed Agents reference material for your project's language.
<code>/claude-in-chrome [task]</code>	Skill	Have Claude carry out a task in your browser, such as testing a page, filling a form, or reading console logs, through Claude in Chrome.
<code>/clear [name]</code>	Built-in	Start a new conversation with empty context. Alias: <code>/reset</code> , <code>/new</code> .
<code>/code-review [low medium high xhigh max ultra] [--fix] [--comment] [--max-findings n all default] [pr# branch path]</code>	Skill	Review the current diff, or a PR number, branch, or path you pass, for correctness bugs. Alias: <code>/review</code> .
<code>/color [color default]</code>	Built-in	Set the prompt bar color for the current session.
<code>/compact [instructions]</code>	Built-in	Free up context by summarizing the conversation so far.
<code>/config [key=value ...]</code>	Built-in	Open the Settings interface to adjust theme, model, output style, and other preferences. Alias: <code>/settings</code> .
<code>/context [all]</code>	Built-in	Visualize current context usage as a colored grid.

<code>/copy [N]</code>	Built-in	Copy the last assistant response to clipboard.
<code>/cost</code>	Built-in	Alias for <code>/usage</code> .
<code>/dataviz [request]</code>	Skill	Design guidance for charts, graphs, and dashboards.
<code>/debug [description]</code>	Skill	Enable debug logging for the current session and troubleshoot issues by reading the session debug log.
<code>/deep-research <question></code>	Workflow	Fan out web searches on a question, fetch and cross-check sources, and synthesize a cited report.
<code>/design [brief]</code>	Skill	Draft UI mockups, screen flows, landing pages, or posters as artboards on one canvas, published as a Claude Design artifact, for example <code>/design a settings screen for a mobile banking app</code> .
<code>/design-login</code>	Built-in	Authorize design-system access for <code>/design-sync</code> with your claude.ai account.
<code>/design-sync [hint]</code>	Skill	Convert your repo's React design system and upload it to Claude Design, so designs it produces use your real components.
<code>/desktop</code>	Built-in	Continue the current session in the Claude Code Desktop app. Alias: <code>/app</code> .
<code>/diff</code>	Built-in	Review the changes in your working tree, including the edits Claude has made so far.
<code>/doctor [prompt-audit [path]]</code>	Skill	Run a setup checkup that diagnoses installation, settings, extension, and <code>CLAUDE.md</code> problems and proposes fixes that Claude applies after you confirm. Alias: <code>/checkup</code> .
<code>/effort [level auto status ultracode [on off]]</code>	Built-in	Set the effort level: <code>low</code> to <code>xhigh</code> , <code>max</code> , or <code>auto</code> ; <code>status</code> prints it. <code>ultracode</code> or <code>ultracode on</code> turns ultracode on for the session at the current level, and <code>ultracode off</code> turns it off; the <code>ultracode</code> key...
<code>/exit</code>	Built-in	Exit the CLI. In an attached background session, this detaches and the session keeps running. Alias: <code>/quit</code> .
<code>/export [filename]</code>	Built-in	Export the current conversation as plain text.
<code>/fast [on off]</code>	Built-in	Toggle fast mode on or off.
<code>/feedback [report]</code>	Built-in	Send product feedback about Claude Code.
<code>/fewer-permission-prompts</code>	Skill	Scan your transcripts for common read-only Bash and MCP tool calls, then add a prioritized allowlist to project <code>.claude/settings.json</code> to reduce permission prompts.
<code>/focus</code>	Built-in	Toggle the focus view, which shows only your last prompt, a one-line tool-call summary with edit diffstats, and the final response.
<code>/fork [prompt]</code>	Built-in	Copy the current conversation into a new background session and keep working here.
<code>/goal [condition clear]</code>	Built-in	Set a goal: Claude keeps working across turns until the condition is met or the goal clears for another reason.
<code>/heapdump</code>	Built-in	Write a JavaScript heap snapshot and a memory breakdown to <code>~/Desktop</code> , or your home directory on Linux without a Desktop folder, for diagnosing high memory usage.
<code>/help</code>	Built-in	Show help and available commands.
<code>/hooks</code>	Built-in	View hook configurations.
<code>/ide</code>	Built-in	Manage IDE integrations and show status.

<code>/import [codex gemini cursor] [--dry-run] [--yes]</code>	Built-in	Bring configuration from OpenAI Codex, Google Gemini CLI, or Cursor on your machine into Claude Code, including instruction files, MCP servers, commands, subagents, and skills.
<code>/init</code>	Built-in	Initialize project with a <code>CLAUDE.md</code> guide.
<code>/insights</code>	Built-in	Generate an HTML report analyzing your recent sessions on this machine: which projects you work in, how you use Claude Code, where things go wrong, and features to try.
<code>/install-github-app</code>	Built-in	Install the Claude GitHub App for a repository, with an optional step to set up GitHub Actions workflows and secrets.
<code>/install-slack-app</code>	Built-in	Install the Claude Slack app.
<code>/keybindings</code>	Built-in	Open your keyboard shortcuts file.
<code>/list-agents</code>	Built-in	List the subagents, agent team teammates, and other Claude Code sessions Claude can message, with the name to use for each.
<code>/login</code>	Built-in	Sign in to your Anthropic account.
<code>/logout</code>	Built-in	Sign out from your Anthropic account.
<code>/loop [interval] [prompt]</code>	Skill	Run a prompt repeatedly while the session stays open. Alias: <code>/proactive</code> .
<code>/mcp [reconnect (<server> all) enable disable [<server> all]]</code>	Built-in	Manage MCP server connections and OAuth authentication.
<code>/memory</code>	Built-in	Edit <code>CLAUDE.md</code> files, enable or disable auto memory, and view auto memory entries.
<code>/mobile</code>	Built-in	Show QR code to download the Claude mobile app. Alias: <code>/ios</code> , <code>/android</code> .
<code>/model [model]</code>	Built-in	Switch the AI model and save it as your default for new sessions.
<code>/output-style [style]</code>	Built-in	List output styles or switch to one, for example <code>/output-style concise</code> .
<code>/passes</code>	Built-in	Share a free week of Claude Code with friends.
<code>/permissions</code>	Built-in	Manage allow, ask, and deny rules for tool permissions. Alias: <code>/allowed-tools</code> .
<code>/plan [description]</code>	Built-in	Enter plan mode directly from the prompt.
<code>/plugin [subcommand]</code>	Built-in	Manage Claude Code plugins.
<code>/plugin-authoring</code>	Built-in	Load the reference Claude works from to write a mod.
<code>/powerup</code>	Built-in	Discover Claude Code features through quick interactive lessons with animated demos.

<code>/pr-comments [PR]</code>	Built-in	Removed in v2.1.91. Ask Claude directly to view pull request comments instead.
<code>/privacy-settings</code>	Built-in	View and update your privacy settings.
<code>/radio</code>	Built-in	Open Claude FM lo-fi radio in your browser.
<code>/rate-limit-options</code>	Built-in	Show ways to keep working when a claude.ai usage limit blocks a request: wait and continue automatically when the limit resets, add usage credits, or upgrade your plan.
<code>/recap</code>	Built-in	Generate a one-line summary of the current session on demand.
<code>/release-notes</code>	Built-in	View the changelog in an interactive version picker.
<code>/reload-plugins [--force]</code>	Built-in	Reload all active plugins to apply pending changes without restarting.
<code>/reload-skills</code>	Built-in	Re-scan skill and command directories so skills added or changed on disk during the session become available without restarting.
<code>/remote-control</code>	Built-in	Make this session available for Remote Control from claude.ai. Alias: <code>/rc</code> .
<code>/remote-env</code>	Built-in	Choose the default cloud environment for cloud sessions you start from the CLI.
<code>/rename [name]</code>	Built-in	Rename the current session and show the name on the prompt bar.
<code>/resume [session]</code>	Built-in	Resume a conversation by ID or name, or open the session picker. Alias: <code>/continue</code> .
<code>/review [low medium high xhigh max ultra] [--fix] [--comment] [--max-findings n all default] [pr# branch path]</code>	Built-in	Alias of <code>/code-review</code> : reviews the current diff, or a PR number, branch, or path you pass, such as <code>/review 1234</code> , and takes the same effort levels and flags.
<code>/rewind</code>	Built-in	Rewind the conversation and/or code to a previous point, or summarize from a selected message. Alias: <code>/checkpoint</code> , <code>/undo</code> .
<code>/run</code>	Skill	Launch and drive your project's app to see a change working, not only passing tests.
<code>/run-skill-generator</code>	Skill	Teach <code>/run</code> and <code>/verify</code> how to build, launch, and drive your project's app from a clean environment by writing a per-project skill.
<code>/sandbox</code>	Built-in	Toggle sandbox mode.
<code>/schedule [description]</code>	Built-in	Create, update, list, or run routines, which execute in the cloud. Alias: <code>/routines</code> .
<code>/scroll-speed</code>	Built-in	Adjust mouse wheel scroll speed interactively.
<code>/security-review</code>	Built-in	Analyze the changes on your current branch for security vulnerabilities.

<code>/setup-bedrock</code>	Built-in	Configure Amazon Bedrock authentication, region, and model pins through an interactive wizard.
<code>/setup-vertex</code>	Built-in	Configure Google Cloud's Agent Platform authentication, project, region, and model pins through an interactive wizard.
<code>/simplify [target]</code>	Skill	Review the changed code for cleanup opportunities and apply the fixes.
<code>/skill-doctor</code>	Built-in	Show what each of your skills costs in context and how often it gets used, so you can find skills to turn off.
<code>/skills</code>	Built-in	List available skills.
<code>/slides [brief]</code>	Skill	Make a new presentation as a Claude Slides artifact filled from your brief, for example <code>/slides a quarterly review of the platform team</code> .
<code>/stats</code>	Built-in	Alias for <code>/usage</code> .
<code>/status</code>	Built-in	Open the Settings interface on the Status tab, showing version, model, account, and connectivity.
<code>/statusline</code>	Built-in	Configure Claude Code's status line.
<code>/stickers</code>	Built-in	Order Claude Code stickers.
<code>/stop</code>	Built-in	Stop the background session you're attached to, or the one you send it to as a peek reply; the transcript and any worktree are kept.
<code>/subtask <task></code>	Built-in	Spawn a forked subagent: a background subagent that inherits the full conversation and works on the task while you keep working.
<code>/tasks</code>	Built-in	View and manage background work in the current session, including subagents that have finished.
<code>/team-onboarding</code>	Built-in	Generate a team onboarding guide from your Claude Code usage history.
<code>/teleport</code>	Built-in	Pull a cloud session into this terminal.
<code>/terminal-setup</code>	Built-in	Install a Shift+Enter keybinding for newlines in VS Code, Cursor, Devin Desktop, Alacritty, or Zed.
<code>/theme</code>	Built-in	Change the color theme.
<code>/tui [default fullscreen]</code>	Built-in	Set the terminal UI renderer and relaunch into it with your conversation intact. <code>fullscreen</code> enables the flicker-free alt-screen renderer.
<code>/ultraplan <prompt></code>	Built-in	Removed.
<code>/ultrareview [PR or branch]</code>	Built-in	Run a deep, multi-agent code review in a cloud sandbox with ultrareview.
<code>/update-config [request]</code>	Skill	Describe a settings change, such as allowing a command, setting an environment variable, or adding a hook, and Claude edits the matching <code>settings.json</code> file.
<code>/upgrade</code>	Built-in	Open the upgrade page in your browser to switch to a higher plan tier.
<code>/usage</code>	Built-in	Show session cost, plan usage limits, and activity stats.

<code>/usage-credits</code>	Built-in	Configure usage credits, or request them from your admin, when you hit a limit.
<code>/verify</code>	Skill	Confirm a code change does what it should by building your project's app, running it, and observing the result, rather than relying on tests or type checks.
<code>/vim</code>	Built-in	Removed in v2.1.92. To toggle between Vim and Normal editing modes, use <code>/config</code> → Editor mode.
<code>/voice</code> <code>[hold tap off]</code>	Built-in	Toggle voice dictation, or enable it in a specific mode.
<code>/web-setup</code>	Built-in	Connect your GitHub account for cloud sessions using your local <code>gh</code> CLI credentials.
<code>/workflow-authoring</code>	Skill	Load the reference for writing dynamic workflow scripts: the script API, resume behavior, quality patterns, and worked examples.
<code>/workflows</code>	Built-in	Open the workflow progress view to watch, pause, resume, or save running and completed workflows.

i NOTE

A few rows say “Removed.” They’re kept here because older tutorials still mention them. For example, `/vim` was replaced by **Editor mode** in `/config`.

Command-Line Flags and Shortcuts Reference

This chapter is the companion to the slash-command reference. It covers what you can type **in your terminal before** Claude Code starts, and every keyboard shortcut you can press **inside** it. The tables were generated from Anthropic's CLI reference (code.claude.com/docs/en/cli-reference) and interactive-mode page (code.claude.com/docs/en/interactive-mode) in October 2026.

IN THIS CHAPTER

- Every `claude ...` subcommand
- Every startup flag, with the ones beginners need marked below
- Every keyboard shortcut and line-editing key

14.1 The ten flags beginners actually use

FLAG	WHY YOU'D USE IT
<code>-c</code> / <code>--continue</code>	Pick up exactly where you left off in this folder
<code>-r</code> / <code>--resume</code>	Choose an older conversation
<code>-p</code> / <code>--print</code>	Ask one question, print the answer, exit, which is great for scripts
<code>--model</code>	Pick the model for this session only
<code>--permission-mode plan</code>	Start in plan mode, so nothing gets edited until you approve
<code>--add-dir ../other</code>	Let Claude also work in a second folder
<code>--effort high</code>	Ask for deeper thinking this session
<code>--debug</code>	Write a detailed log when something misbehaves
<code>--chrome</code>	Turn on browser control for web testing
<code>--dangerously-skip-permissions</code>	Skip all prompts. Only use it in a throwaway container or VM (see Vol. 8)

⚠ DANGER

Flags with “dangerously” in their name remove safety checks. On your everyday computer, use plan mode, accept-edits mode, or allow rules instead.

14.2 Terminal subcommands

COMMAND	WHAT IT DOES
<code>claude</code>	Start interactive session.
<code>claude "query"</code>	Start interactive session with initial prompt.
<code>claude -p "query"</code>	Query via SDK, then exit.
<code>claude -c</code>	Continue most recent conversation in current directory.
<code>claude -c -p "query"</code>	Continue via SDK.
<code>claude -r "<session>" "query"</code>	Resume session by ID or name.
<code>claude update</code>	Update to latest version.
<code>claude gateway</code>	Start the self-hosted Claude apps gateway server, for administrators deploying SSO and policy in front of Claude Code on Amazon Bedrock, Google Cloud's Agent Platform, or Microsoft Foundry.
<code>claude install [version]</code>	Install or reinstall the native binary.
<code>claude auth login</code>	Sign in to your Anthropic account.

<code>claude auth logout</code>	Log out from your Anthropic account.
<code>claude auth status</code>	Show authentication status as JSON. Use <code>--text</code> for human-readable output.
<code>claude agents</code>	Open agent view to monitor and dispatch parallel background sessions.
<code>claude attach <id name></code>	Attach to a background session in this terminal.
<code>claude auto-mode defaults</code>	Print the built-in auto mode classifier rules as JSON. Use <code>claude auto-mode config</code> to see your effective config with settings applied. <code>--label <prefix></code> prints only the rules whose label starts...
<code>claude auto-mode reset</code>	Restore the default auto mode configuration by removing the <code>autoMode</code> section from your user settings file.
<code>claude daemon logs</code>	Follow the background-session supervisor's log file, <code>~/.claude/daemon.log</code> , printing new lines as they arrive until you press <code>Ctrl+C</code> .
<code>claude daemon run</code>	Run the background-session supervisor in the foreground of this terminal, printing its log.
<code>claude daemon status</code>	Print the background-session supervisor's state, version, socket directory, and worker count for diagnostics.
<code>claude daemon stop --any</code>	Stop the background-session supervisor and the sessions it hosts.
<code>claude doctor</code>	Print read-only installation and settings diagnostics from the terminal without starting a session, including install health, settings-file validation errors, and Remote Control eligibility.
<code>claude import [source]</code>	Start an interactive session that runs <code>/import</code> to bring configuration from other coding agents into Claude Code.
<code>claude logs <id name></code>	Print recent output from a background session.
<code>claude mcp</code>	Configure Model Context Protocol (MCP) servers.
<code>claude mcp login <name></code>	Run a configured MCP server's OAuth flow without opening the interactive <code>/mcp</code> panel.
<code>claude mcp logout <name></code>	Clear stored OAuth credentials for an MCP server.
<code>claude plugin</code>	Manage Claude Code plugins.
<code>claude purge [path]</code>	Delete all local Claude Code state for a project: transcripts, task lists, debug logs, file-edit history, prompt history lines, and the project's entry in <code>~/.claude.json</code> .
<code>claude remote- control</code>	Start a Remote Control server to control Claude Code from Claude.ai or the Claude app.
<code>claude respawn <id></code>	Restart a background session, running or stopped, with its conversation intact.
<code>claude rm <id></code>	Remove a background session from the list.
<code>claude self- hosted-runner</code>	Start a runner process that registers this machine or container with a self-hosted environment and hosts Claude Code cloud sessions on your infrastructure.
<code>claude setup- token</code>	Generate a long-lived OAuth token for CI and scripts.
<code>claude stop <id></code>	Stop a background session.
<code>claude ultrareview [target]</code>	Run ultrareview non-interactively.

14.3 Every startup flag

FLAG	WHAT IT DOES
<code>--add-dir</code>	Add additional working directories for Claude to read and edit files.
<code>--advisor <model></code>	Enable the server-side advisor tool for this session with a model alias, <code>fable</code> , <code>opus</code> , or <code>sonnet</code> , or a full model ID. Takes precedence over the <code>advisorModel</code> ...
<code>--agent</code>	Specify an agent for the current session (overrides the <code>agent</code> setting).
<code>--agents</code>	Define custom subagents dynamically via JSON. Accepts the fields listed for CLI-defined subagents.
<code>--allow-dangerously-skip-permissions</code>	Add <code>bypassPermissions</code> to the <code>Shift+Tab</code> mode cycle without starting in it.
<code>--allowedTools</code> , <code>--allowed-tools</code>	Tools that execute without prompting for permission, apart from reads from network paths.
<code>--append-subagent-system-prompt</code>	Append custom text to the end of every subagent's system prompt, nested subagents included, apart from a forked subagent, which reuses the conversation's own prompt.
<code>--append-subagent-system-prompt-file</code>	Load text from a file and append it to subagent system prompts.
<code>--append-system-prompt</code>	Append custom text to the end of the default system prompt.
<code>--append-system-prompt-file</code>	Load additional system prompt text from a file and append to the default prompt.
<code>--autocompact <auto tokens></code>	Set the auto-compact window for this session without changing your saved settings.
<code>--ax-screen-reader</code>	Render screen-reader friendly output: flat text without decorative borders or animations.
<code>--bare</code>	Minimal mode: skip auto-discovery of hooks, skills, custom commands, subagents, installed plugins, MCP servers, auto memory, and CLAUDE.md so scripted calls start faster.
<code>--betas</code>	Beta headers to include in API requests (API key users only).
<code>--bg</code> , <code>--background</code>	Start the session as a background agent and return immediately.
<code>--channels</code>	(Research preview) MCP servers whose channel notifications Claude should listen for in this session.
<code>--chrome</code>	Enable Chrome browser integration for web automation and testing.
<code>--cloud</code>	With a task description, create a new cloud session.
<code>--continue</code> , <code>-c</code>	Load the most recent conversation in the current directory, including a background session that has finished; opening finished background sessions requires Claude...
<code>--dangerously-load-development-channels</code>	Enable channels that are not on the approved allowlist, for local development.
<code>--dangerously-skip-permissions</code>	Skip permission prompts.
<code>--debug</code>	Enable debug mode with optional category filtering, such as <code>--debug='mcp, startup'</code> or <code>--debug='!lp'</code> .
<code>--debug-file <path></code>	Write debug logs to a specific file path.

<code>--desktop</code>	Open the Claude Desktop app on the current directory and exit without starting a session in the terminal.
<code>--disable-slash-commands</code>	Disable all skills and commands for this session.
<code>--disallowedTools</code> , <code>--disallowed-tools</code>	Deny rules.
<code>--effort</code>	Set the effort level for the current session.
<code>--enable-auto-mode</code>	Removed in v2.1.111. Auto mode is now in the <code>Shift+Tab</code> cycle by default; use <code>--permission-mode auto</code> to start in it.
<code>--environment <environment-id></code>	Create a new cloud session that runs on the self-hosted environment with the given ID. Environment IDs start with <code>ccpool_</code> .
<code>--exclude-dynamic-system-prompt-sections</code>	Move per-user context, such as the auto memory location, out of the system prompt and into the first user message.
<code>--exec</code>	Run a shell command as a PTY-backed background job instead of starting a Claude session.
<code>--fallback-model</code>	Enable automatic fallback to the specified model(s) when the primary model is overloaded or not available, for example a retired model.
<code>--fork-session</code>	When resuming, create a new session ID instead of reusing the original (use with <code>--resume</code> or <code>--continue</code>).
<code>--forward-subagent-text</code>	Emit subagent text and thinking blocks in the output stream as <code>assistant</code> and <code>user</code> messages with <code>parent_tool_use_id</code> set, so you can reconstruct each subagent's...
<code>--from-pr</code>	Open the session picker filtered to sessions linked to a specific pull request.
<code>--ide</code>	Automatically connect to IDE on startup if exactly one valid IDE is available.
<code>--init</code>	Run Setup hooks with the <code>init</code> matcher before the session (print mode only).
<code>--init-only</code>	Run Setup and <code>SessionStart</code> hooks, then exit without starting a conversation.
<code>--include-hook-events</code>	Include hook lifecycle events in the output stream. <code>SessionStart</code> and <code>Setup</code> hook events are always included and don't need this flag.
<code>--include-partial-messages</code>	Include partial streaming events in output.
<code>--input-format</code>	Specify input format for print mode (options: <code>text</code> , <code>stream-json</code>).
<code>--json-schema</code>	Get validated JSON output matching a JSON Schema after the agent completes its workflow (print mode only).
<code>--maintenance</code>	Run Setup hooks with the <code>maintenance</code> matcher before the session (print mode only).
<code>--max-budget-usd</code>	Stop the run once estimated spend on API calls reaches this amount (print mode only).
<code>--max-turns</code>	Limit the number of agentic turns (print mode only).
<code>--mcp-config</code>	Load MCP servers from JSON files or strings (space-separated).
<code>--model</code>	Sets the model for the current session with a model alias such as <code>sonnet</code> , <code>opus</code> , <code>haiku</code> , or <code>fable</code> , or a model's full name.
<code>--name</code> , <code>-n</code>	Set a display name for the session, shown in <code>/resume</code> and the terminal title.
<code>--no-chrome</code>	Disable Chrome browser integration for this session.
<code>--no-session-persistence</code>	Disable session persistence so sessions are not saved to disk and cannot be resumed.
<code>--output-format</code>	Specify output format for print mode (options: <code>text</code> , <code>json</code> , <code>stream-json</code>).
<code>--permission-mode</code>	Begin in a specified permission mode.

<code>--permission-prompt-tool</code>	Specify an MCP tool to handle permission prompts in non-interactive mode.
<code>--permission-prompts</code>	Set who answers permission prompts in print mode.
<code>--plugin-dir</code>	Load a plugin from a directory or <code>.zip</code> archive, or several from a folder of plugins, for this session only.
<code>--plugin-url</code>	Fetch a plugin <code>.zip</code> archive from a URL for this session only.
<code>--print, -p</code>	Print response without interactive mode (see Agent SDK documentation for programmatic usage details).
<code>--prompt-suggestions</code>	Emit a <code>prompt_suggestion</code> message with a predicted next user prompt after each turn that generates one; very short conversations can produce none.
<code>--ref <branch></code>	With <code>--environment</code> , base the new session's checkout on a named ref instead of local <code>HEAD</code> .
<code>--remote</code>	Deprecated alias for <code>--cloud</code> , including the existing-session form.
<code>--remote-control, --rc</code>	Start an interactive session with Remote Control enabled so you can also control it from claude.ai or the Claude app.
<code>--remote-control-session-name-prefix <prefix></code>	Prefix for auto-generated Remote Control session names when no explicit name is set.
<code>--replay-user-messages</code>	Re-emit user messages from stdin back on stdout for acknowledgment.
<code>--restricted</code>	Start in restricted mode.
<code>--resume, -r</code>	Resume a specific session by ID or name, or show an interactive picker to choose a session.
<code>--safe-mode</code>	Start with all customizations disabled to troubleshoot a broken configuration: CLAUDE.md, skills, plugins, hooks, MCP servers, custom commands and agents, output...
<code>--session-id</code>	Use a specific session ID for the conversation (must be a valid UUID).
<code>--setting-sources</code>	Comma-separated list of setting sources to load (<code>user</code> , <code>project</code> , <code>local</code>).
<code>--settings</code>	Path to a settings JSON file or an inline JSON string.
<code>--strict-mcp-config</code>	Only use MCP servers from <code>--mcp-config</code> , ignoring all other MCP configurations.
<code>--system-prompt</code>	Replace the entire system prompt with custom text.
<code>--system-prompt-file</code>	Load system prompt from a file, replacing the default prompt.
<code>--system-prompt-snapshot</code>	Pass <code>off</code> to rebuild the system prompt on every request instead of reusing the prompt recorded on the conversation's first request, for example while you iterate on...
<code>--teleport</code>	Resume a cloud session in your local terminal.
<code>--teammate-mode</code>	Set how agent team teammates display: <code>in-process</code> (default), <code>auto</code> , <code>tmux</code> , or <code>iterm2</code> .
<code>--tmux</code>	Create a tmux session for the worktree.
<code>--tools</code>	Restrict which built-in tools Claude can use.
<code>--verbose</code>	Enable verbose logging, shows full turn-by-turn output.
<code>--version, -v</code>	Output the version number.
<code>--worktree, -w</code>	Start Claude in an isolated git worktree at <code><repo>/claude/worktrees/<name></code> .
<code>--system-prompt</code>	Replaces the entire default prompt.
<code>--system-prompt-file</code>	Replaces with file contents.
<code>--append-system-prompt</code>	Appends to the default prompt.

<code>--append-system-prompt-file</code>	Appends file contents to the default prompt.
<code>--system-prompt-snapshot</code>	With <code>off</code> , rebuilds the prompt on every request.

14.4 Every keyboard shortcut

KEYS	WHAT IT DOES
<code>Ctrl+C</code>	Interrupt, or clear input.
<code>Ctrl+X Ctrl+K</code>	Stop all running background subagents in this session, and turn off artifact auto-replies for the rest of it.
<code>Ctrl+D</code>	Exit Claude Code session.
<code>Ctrl+G</code> or <code>Ctrl+X Ctrl+E</code>	Open in default text editor.
<code>Ctrl+L</code>	Redraw the screen.
<code>Ctrl+O</code>	Toggle transcript viewer.
<code>Ctrl+R</code>	Reverse search command history.
<code>Ctrl+V</code> or <code>Cmd+V</code> (iTerm2) or <code>Alt+V</code> (Windows and WSL)	Paste image from clipboard.
<code>Ctrl+B</code>	Background running tasks.
<code>Ctrl+T</code>	Toggle Claude's task checklist.
<code>Ctrl+S</code>	Stash or restore prompt.
<code>Ctrl+Z</code>	Suspend Claude Code.
<code>Left/Right arrows</code>	Cycle through dialog tabs.
<code>Tab</code>	Accept an autocomplete suggestion, or add a comment to a permission answer.
<code>Up/Down arrows</code> or <code>Ctrl+P</code> / <code>Ctrl+N</code>	Move cursor or navigate command history.
<code>Esc</code>	Interrupt Claude, or close a dialog.
<code>Esc + Esc</code>	Clear input draft, or rewind.
<code>Ctrl+Enter</code> or <code>Ctrl+X Ctrl+S</code>	Send queued messages now.
<code>Shift+Tab</code> , or <code>Alt+M</code> on Windows when the Node or Bun runtime doesn't enable VT input mode	Cycle permission modes.
<code>Option+P</code> (macOS) or <code>Alt+P</code> (Windows/Linux)	Switch model.
<code>Option+T</code> (macOS) or <code>Alt+T</code> (Windows/Linux)	Toggle extended thinking.
<code>Option+O</code> (macOS) or <code>Alt+O</code> (Windows/Linux)	Toggle fast mode.
<code>Ctrl+A</code>	Move cursor to start of current line.
<code>Ctrl+E</code>	Move cursor to end of current line.
<code>Ctrl+K</code>	Delete to end of line.
<code>Ctrl+U</code>	Delete from cursor to line start.
<code>Ctrl+W</code>	Delete back to previous whitespace.
<code>Ctrl+Y</code>	Paste deleted text.
<code>Alt+Y</code> (after <code>Ctrl+Y</code>)	Cycle paste history.

Alt+B	Move cursor back one word.
Alt+F	Move cursor forward one word.
Alt+D	Delete to end of word.
Ctrl+_ or Ctrl+Shift+-	Undo last input edit.
Ctrl+T	Toggle syntax highlighting for code blocks.
/ at start	Command or skill.
! at start	Shell mode.
@	File path mention.
:	Emoji shortcode.
? on empty input	Toggle the shortcut help panel.
?	Toggle the keyboard shortcut help panel.
{ / }	Jump to the previous or next user prompt, like vim paragraph motion.
Ctrl+E	Toggle show all content.
[	Write the full conversation to your terminal's native scrollbar so Cmd+F , tmux copy mode, and other native tools...
v	Write the conversation to a temporary file and open it in \$VISUAL or \$EDITOR .
q , Ctrl+C , Esc	Exit transcript view.
Esc or Ctrl+[	Enter NORMAL mode.
i	Insert before cursor.
I	Insert at beginning of line.
a	Insert after cursor.
A	Insert at end of line.
o	Open line below.
O	Open line above.
v	Start character-wise visual selection.
V	Start line-wise visual selection.
h / j / k / l	Move left/down/up/right.
Space	Move right.
w	Next word.
e	End of word.
b	Previous word.
0	Beginning of line.
\$	End of line.
^	First non-blank character.
gg	Beginning of input.

G	Beginning of last line.
f{char}	Jump to next occurrence of character on the current line.
F{char}	Jump to previous occurrence of character on the current line.
t{char}	Jump to just before next occurrence of character on the current line.
T{char}	Jump to just after previous occurrence of character on the current line.
;	Repeat last f/F/t/T motion.
,	Repeat last f/F/t/T motion in reverse.
%	Jump to the matching () , [] , or { } bracket, starting from the next bracket on the current line when the cursor...
/	Open reverse history search, same as Ctrl+R .
x	Delete character.
r{char}	Replace character under cursor with {char} .
dd	Delete line.
D	Delete to end of line.
dw / de / db	Delete word/to end/back.
df{char} / dt{char}	Delete to and including, or up to, the next occurrence of a character on the current line.
dj / dk	Delete the current line and the line below or above.
dgg / dG	Delete from the current line to the first or last line.
d0 / c0 / y0	Delete, change, or yank from the cursor back to the beginning of the line.
d% / c% / y%	Delete, change, or yank from the cursor through the bracket that % jumps to.
cc	Change line.
C	Change to end of line.
cw / ce / cb	Change word/to end/back.
s	Substitute character: delete the character under the cursor and enter INSERT mode.
S	Substitute line: clear the line and enter INSERT mode.
yy / Y	Yank (copy) line.
yw / ye / yb	Yank word/to end/back.
p	Paste after cursor.
P	Paste before cursor.
>>	Indent line.
<<	Dedent line.
J	Join lines.
u	Undo.
.	Repeat last change.
iw / aw	Inner/around word.
iW / aW	Inner/around WORD (whitespace-delimited).

<code>i" / a"</code>	Inner/around double quotes.
<code>i' / a'</code>	Inner/around single quotes.
<code>i(/ a(</code>	Inner/around parentheses.
<code>i[/ a[</code>	Inner/around brackets.
<code>i{ / a{</code>	Inner/around braces.
<code>d / x</code>	Delete selection.
<code>y</code>	Yank selection.
<code>c / s</code>	Change selection.
<code>p</code>	Replace selection with register contents.
<code>r{char}</code>	Replace every selected character with <code>{char}</code> .
<code>~ / u / U</code>	Toggle, lowercase, or uppercase selection.
<code>> / <</code>	Indent or dedent selected lines.
<code>J</code>	Join selected lines.
<code>o</code>	Swap cursor and anchor.
<code>iw / aw / i" / ...</code>	Select a text object.
<code>v / V</code>	Toggle between character-wise and line-wise, or exit.
<code>Space</code> , <code>Enter</code> , <code>Escape</code>	Dismiss the answer and return to the prompt.
<code>Up</code> / <code>Down</code>	Scroll the answer.
<code>Shift+Left</code> / <code>Shift+Right</code>	Step between this answer and your earlier <code>/btw</code> answers. <code>Shift+Left</code> moves to older answers and <code>Shift+Right</code> ...
<code>c</code>	Copy the answer to your clipboard as raw Markdown.
<code>f</code>	Start a forked subagent that inherits the parent conversation plus this question and answer, so it can continue with...
<code>x</code>	Clear the list of earlier <code>/btw</code> exchanges shown above the current answer.

Real-World Workflows, Step by Step

Knowing the commands is half the job. This chapter strings them into complete, repeatable workflows you can follow with the book open next to your keyboard.

IN THIS CHAPTER

- Fix a bug from an error message
- Add a feature safely with plan mode
- Understand a project you've never seen
- Write and run tests
- Ship a pull request
- Run long tasks in the background
- Automate with `claude -p`

15.1 Workflow 1: Fix a bug from an error message

1

Reproduce it.

Run the program and copy the full error, including the lines above it.

2

Start Claude in the project folder.

Type `claude`.

3 Paste the error with context.**PROMPT**

When I run `npm start` I get the error below. Find the cause, explain it in plain English, then propose the smallest fix. Don't change anything else.
[paste error]

4 Read the explanation before approving.

If it doesn't make sense to you, ask "why?" until it does.

5 Approve the edit, then ask Claude to rerun the command.

A fix isn't done until the error is gone.

6 Commit.

Ask: "Commit this with a clear message."

15.2 Workflow 2: Add a feature with plan mode

1 Enter plan mode.

Press `Shift` + `Tab` until the footer says *plan*, or type `/plan add a dark-mode toggle`.

2 Let Claude research.

It reads files but can't edit them.

3 Review the plan.

Push back on anything too big. Press `Ctrl` + `G` to edit the plan in your text editor.

4 Approve.

Claude switches out of plan mode and starts working.

5 Check in at milestones.

Ask for a short status after each step.

6 Review with `/diff`, then `/code-review`.**7 Compact or clear.**

If the session is long and you're continuing, use `/compact`. If you're done, commit and `/clear`.

15.3 Workflow 3: Learn an unfamiliar project

PROMPT

Give me a beginner-friendly tour of this project: what it does, how to run it, the main folders, and the five files I should read first. Don't edit anything.

Follow up with “trace what happens when a user clicks *Sign up*.” This **trace** question is the fastest way to learn real code. When you're done, run `/init` so the next session starts with what you learned.

15.4 Workflow 4: Tests first

- 1 **Ask for tests that describe the behaviour you want**, and tell Claude they should fail for now.
- 2 **Run them and confirm they fail.**
This proves the tests actually check something.
- 3 **Ask Claude to make the tests pass without changing the tests.**
- 4 **Run the full test suite** so you know nothing else broke.

TIP

Saying “don't modify the tests” stops a common shortcut, where an AI “fixes” a failing test by weakening it.

15.5 Workflow 5: Ship a pull request

- 1 **Branch first.**
“Create a branch called `feature/contact-form`.”
- 2 **Do the work** (Workflow 2).
- 3 **Review.**
`/diff` → `/code-review` → `/security-review`.
- 4 **Commit and push.**
“Commit in logical chunks and push the branch.”
- 5 **Open the PR.**
“Open a pull request with a summary and testing notes.” This uses the `gh` tool (Vol. 2).

6

Address feedback.

Paste reviewer comments back to Claude, or try `/autofix-pr` if your plan supports cloud sessions.

15.6 Workflow 6: Long tasks in the background

Press `Ctrl + B` to send a long-running command, like a dev server or test run, to the background. Use `/tasks` to check on it. To detach the whole session and get your terminal back, use `/background`. Then watch it from your shell with `claude agents`. `/subtask` hands a side job to a helper that sees the full conversation.

15.7 Workflow 7: Scripts with `claude -p`

`-p` (print mode) runs one request with no interactive screen. That makes it ideal for scripts and scheduled jobs.

TERMINAL

```
claude -p "Summarize what changed in git today in five bullets" > today.md
git diff | claude -p "Write a commit message for this diff"
```

⚠ WARNING

In print mode nobody is around to approve prompts. Give it only the permissions it needs, with `--allowedTools` or allow rules, and never broad shell access on a machine you care about.

15.8 Daily routine checklist

- ☐ `git status` is clean before you start
- ☐ Start with `claude -c` to continue, or `claude` for a fresh task
- ☐ One task per conversation
- ☐ Use plan mode for anything touching more than two files
- ☐ `/context` when things feel slow, then `/compact` or `/clear`
- ☐ `/diff` before every commit
- ☐ Commit small and often
- ☐ End by asking for a summary of what changed and what's next

Troubleshooting and FAQ

When something goes wrong, work through this chapter from top to bottom. Most problems are one of a dozen common causes.

IN THIS CHAPTER

- A first-aid routine for any problem
- Install and sign-in fixes
- When Claude gets slow, forgetful, or stuck
- Permission and tool problems
- Frequently asked questions

16.1 First-aid routine

- 1 Read the whole message.**
The fix is often in the last line.
- 2 Run `claude doctor` in your terminal, or `/doctor` inside a session.**
- 3 Update.**
Run `claude update`, then restart.
- 4 Check sign-in.**
Run `claude auth status`.
- 5 Turn on logging.**
Restart with `claude --debug`, or run `/debug describe the problem`.
- 6 Search the official docs at code.claude.com/docs.**
- 7 Report it.**
If it's a real bug, use `/bug`. It shows you what will be sent and asks you to confirm first.

16.2 Symptom table

SYMPTOM	LIKELY CAUSE	FIX
<code>command not found: claude</code>	The install folder isn't on your PATH	Open a new terminal. Run <code>claude doctor</code> . Reinstall with the native installer
Browser sign-in never finishes	Pop-up blocked or wrong browser profile	Copy the sign-in URL into the right browser. Try <code>claude auth login</code>
"Usage limit reached"	Your plan's limit for this period is used up	<code>/usage</code> to see when it resets. <code>/rate-limit-options</code> lists choices. Switch to a lighter <code>/model</code>
Answers get vague or repetitive	The context window is full of old material	<code>/context</code> , then <code>/compact</code> or <code>/clear</code>
Claude keeps redoing a mistake	The bad attempt is still in context	<code>/rewind</code> to before the mistake. Re-prompt with the missing detail
Too many permission prompts	No allow rules yet	<code>/permissions</code> to add rules, or <code>/fewer-permission-prompts</code>
Claude edits files you didn't want touched	The request was too vague	Use plan mode. Add "only change X" to the prompt. Add deny rules
MCP server shows "failed"	Bad config or expired sign-in	<code>/mcp</code> to reconnect or sign in. Check the command and path
A skill doesn't appear in <code>/</code>	Wrong folder, bad YAML, or a name clash	Check <code>.claude/skills/<name>/SKILL.md</code> . Run <code>/reload-skills</code>
Hooks don't fire	Wrong event name or matcher	Run <code>/hooks</code> to inspect them, then <code>--debug</code>
Garbled screen	The terminal display got out of sync	<code>Ctrl</code> + <code>L</code> to redraw
Pasting an image does nothing	Your terminal uses a different key	Try <code>Ctrl</code> + <code>V</code> , <code>Cmd</code> + <code>V</code> (iTerm2), or <code>Alt</code> + <code>V</code> (Windows)
<code>Shift</code> + <code>Enter</code> sends instead of adding a new line	Your terminal doesn't support it	Run <code>/terminal-setup</code> , or use <code>\</code> + <code>Enter</code> or <code>Ctrl</code> + <code>J</code>

16.3 Frequently asked questions

Does `/clear` delete my files? No. It only clears the conversation. Your files and git history stay as they are.

Is `/compact` the same as `/clear`? No. `/compact` keeps a summary and stays on the same task. `/clear` starts from nothing.

Can Claude see my whole computer? By default it works in the folder you started it in, plus any folders you add with `/add-dir`. It asks before it does anything outside its allowed permissions.

Does it remember me between sessions? Only through `CLAUDE.md`, auto memory, and the conversations you resume. A fresh session doesn't automatically know what you discussed yesterday.

What does it cost? It depends on your plan and changes over time. See the Costs chapter, and run `/usage` for your own numbers. This manual doesn't quote prices, because they change.

Is my code used for training? That depends on your account type and settings. Check Anthropic's current privacy settings for your plan. Don't take a third-party guide's word for it.

Can I use it on my phone? You can use Claude Code on the web and in the Claude app, and `/remote-control` lets you drive a session running on your computer. See the "Beyond the Terminal" chapter.

Hooks, Settings, and Environment Reference

This is the deep reference for people who want to automate Claude Code. You don't need it on day one. The tables were generated from Anthropic's hooks reference (code.claude.com/docs/en/hooks) and environment-variable reference (code.claude.com/docs/en/env-vars) in October 2026.

IN THIS CHAPTER

- Every hook event and when it fires
- A worked hook example
- Where settings files live and which one wins
- Every documented environment variable

17.1 Settings files and precedence

FILE	SCOPE	COMMIT TO GIT?
Managed settings (set by IT)	Whole organisation, can't be overridden	Not applicable
<code>.claude/settings.local.json</code>	You, in this project	No, keep it private
<code>.claude/settings.json</code>	Everyone on this project	Yes
<code>~/.claude/settings.json</code>	You, in every project	Not applicable

When two files disagree, the more specific one wins: managed, then command-line flags, then local project, then shared project, then user. **Deny** rules beat allow rules wherever they come from.

17.2 Hook events

A **hook** is a command that runs automatically when an event fires. Use `/hooks` to see the hooks you have, or `/update-config` to ask Claude to add one.

EVENT	FIRES...
<code>SessionStart</code>	When a session begins or resumes.
<code>Setup</code>	When you start Claude Code with <code>--init-only</code> , or with <code>--init</code> or <code>--maintenance</code> in <code>-p</code> mode.
<code>UserPromptSubmit</code>	When a prompt is submitted, before Claude processes it.
<code>UserPromptExpansion</code>	When a user-typed command expands into a prompt, before it reaches Claude.
<code>PreToolUse</code>	Before a tool call executes.
<code>PermissionRequest</code>	When a tool call needs a permission decision.
<code>PermissionDenied</code>	When auto mode denies a tool call, including denials without a classifier verdict.
<code>PostToolUse</code>	After a tool call succeeds.
<code>PostToolUseFailure</code>	After a tool call fails.
<code>PostToolBatch</code>	After a full batch of parallel tool calls resolves, before the next model call.
<code>Notification</code>	When Claude Code sends a notification.
<code>MessageDisplay</code>	While assistant message text is displayed.
<code>SubagentStart</code>	When a subagent is spawned.
<code>SubagentStop</code>	When a subagent finishes.
<code>TaskCreated</code>	When a task is being created via <code>TaskCreate</code> .
<code>TaskCompleted</code>	When a task is being marked as completed.
<code>Stop</code>	When Claude finishes responding.
<code>StopFailure</code>	When the turn ends due to an API error.
<code>TeammateIdle</code>	When an agent team teammate is about to go idle.
<code>InstructionsLoaded</code>	When a <code>CLAUDE.md</code> or <code>.claude/rules/*.md</code> file is loaded into context.
<code>ConfigChange</code>	When a configuration file changes during a session.
<code>CwdChanged</code>	When the working directory changes, for example when Claude executes a <code>cd</code> command.
<code>DirectoryAdded</code>	When a working directory is added mid-session via <code>/add-dir</code> or the SDK <code>register_repo_root</code> control request.
<code>FileChanged</code>	When a watched file changes on disk.
<code>WorktreeCreate</code>	When a worktree is being created via <code>--worktree</code> , <code>isolation: "worktree"</code> , or for a background session.
<code>WorktreeRemove</code>	When a worktree that a <code>WorktreeCreate</code> hook created is being removed.
<code>PreCompact</code>	Before context compaction.
<code>PostCompact</code>	After context compaction completes.
<code>PreModelSwitch</code>	Before Claude Code applies a model switch that you or a client requested.

PostModelSwitch	After the session's model changes, including changes Claude Code makes on its own, such as restoring the model when you resume a session.
Elicitation	When an MCP server requests user input during a tool call.
ElicitationResult	After a user responds to an MCP elicitation, before the response is sent back to the server.
SessionEnd	When a session terminates.

17.3 Worked example: format every file Claude edits

```
.CLAUDE/SETTINGS.JSON

{
  "hooks": {
    "PostToolUse": [
      {
        "matcher": "Edit|Write",
        "hooks": [
          { "type": "command", "command": "jq -r '.tool_input.file_path' | xargs npx prettier --write"
        ]
      }
    ]
  }
}
```

⚠ WARNING

Hooks run with **your** user permissions and no prompt. Only add hooks you've read and understand, and be careful with hooks that come inside repositories you cloned.

The hook receives details of the tool call as JSON on its input. `jq` pulls out the edited file's path, so Prettier formats only that file. You need `jq` and Prettier installed.

17.4 Environment variables

Set these in your shell or in the `env` block of `settings.json`. Most people never need more than a handful. Search this table when a guide or error message mentions one.

VARIABLE	WHAT IT DOES
ANTHROPIC_API_KEY	API key sent as <code>X-API-Key</code> header.
ANTHROPIC_AUTH_TOKEN	Custom value for the <code>Authorization</code> header (the value you set here will be prefixed with <code>Bearer </code>).
ANTHROPIC_AWS_API_KEY	Workspace API key for Claude Platform on AWS, generated in the AWS Console.
ANTHROPIC_AWS_BASE_URL	Override the Claude Platform on AWS endpoint URL. Use for custom regions or when routing through an LLM...

<code>ANTHROPIC_AWS_WORKSPACE_ID</code>	Required for Claude Platform on AWS. Sent on every request as the <code>anthropic-workspace-id</code> header.
<code>ANTHROPIC_BASE_URL</code>	Override the API endpoint to route requests through a proxy or gateway.
<code>ANTHROPIC_BEDROCK_BASE_URL</code>	Override the Amazon Bedrock endpoint URL. Use for custom Amazon Bedrock endpoints or when routing through...
<code>ANTHROPIC_BEDROCK_MANTLE_BASE_URL</code>	Override the Amazon Bedrock Mantle endpoint URL. See Mantle endpoint.
<code>ANTHROPIC_BEDROCK_REGION_PREFIX</code>	Cross-region inference profile prefix (<code>us</code> , <code>eu</code> , <code>apac</code> , <code>jp</code> , <code>au</code> , or <code>global</code>) Claude Code tries...
<code>ANTHROPIC_BEDROCK_SERVICE_TIER</code>	Amazon Bedrock service tier (<code>default</code> , <code>flex</code> , or <code>priority</code>).
<code>ANTHROPIC_BETAS</code>	Comma-separated list of additional <code>anthropic-beta</code> header values to include in API requests.
<code>ANTHROPIC_CUSTOM_HEADERS</code>	Custom headers to add to requests (<code>Name: Value</code> format, newline-separated for multiple headers).
<code>ANTHROPIC_CUSTOM_MODEL_OPTION</code>	Model ID to add as a custom entry in the <code>/model</code> picker.
<code>ANTHROPIC_CUSTOM_MODEL_OPTION_DESCRIPTION</code>	Display description for the custom model entry in the <code>/model</code> picker.
<code>ANTHROPIC_CUSTOM_MODEL_OPTION_NAME</code>	Display name for the custom model entry in the <code>/model</code> picker.
<code>ANTHROPIC_CUSTOM_MODEL_OPTION_SUPPORTED_CAPABILITIES</code>	Comma-separated list of capabilities the custom model supports, for example <code>effort,thinking</code> .
<code>ANTHROPIC_DEFAULT_FABLE_MODEL</code>	Model ID that the <code>fable</code> alias resolves to, and the ID Claude Code recognizes as a Fable model for...
<code>ANTHROPIC_DEFAULT_FABLE_MODEL_DESCRIPTION</code>	Display description for the pinned Fable model in the <code>/model</code> picker.
<code>ANTHROPIC_DEFAULT_FABLE_MODEL_NAME</code>	Display name for the pinned Fable model in the <code>/model</code> picker.
<code>ANTHROPIC_DEFAULT_FABLE_MODEL_SUPPORTED_CAPABILITIES</code>	Comma-separated list of capabilities the pinned Fable model supports, for example <code>effort,thinking</code> .
<code>ANTHROPIC_DEFAULT_HAIKU_MODEL</code>	Model ID that the <code>haiku</code> alias resolves to, also used for background functionality.
<code>ANTHROPIC_DEFAULT_HAIKU_MODEL_DESCRIPTION</code>	Display description for the pinned Haiku model in the <code>/model</code> picker.
<code>ANTHROPIC_DEFAULT_HAIKU_MODEL_NAME</code>	Display name for the pinned Haiku model in the <code>/model</code> picker.
<code>ANTHROPIC_DEFAULT_HAIKU_MODEL_SUPPORTED_CAPABILITIES</code>	Comma-separated list of capabilities the pinned Haiku model supports, for example <code>effort,thinking</code> .
<code>ANTHROPIC_DEFAULT_MODEL</code>	Model that new sessions start on by default.
<code>ANTHROPIC_DEFAULT_OPUS_MODEL</code>	Model ID that the <code>opus</code> alias resolves to, and that <code>opusplan</code> uses while Plan Mode is active.
<code>ANTHROPIC_DEFAULT_OPUS_MODEL_DESCRIPTION</code>	Display description for the pinned Opus model in the <code>/model</code> picker.

<code>ANTHROPIC_DEFAULT_OPUS_MODEL_NAME</code>	Display name for the pinned Opus model in the <code>/model</code> picker.
<code>ANTHROPIC_DEFAULT_OPUS_MODEL_SUPPORTED_CAPABILITIES</code>	Comma-separated list of capabilities the pinned Opus model supports, for example <code>effort,thinking</code> .
<code>ANTHROPIC_DEFAULT_SONNET_MODEL</code>	Model ID that the <code>sonnet</code> alias resolves to, and that <code>opusplan</code> uses when Plan Mode is not active.
<code>ANTHROPIC_DEFAULT_SONNET_MODEL_DESCRIPTION</code>	Display description for the pinned Sonnet model in the <code>/model</code> picker.
<code>ANTHROPIC_DEFAULT_SONNET_MODEL_NAME</code>	Display name for the pinned Sonnet model in the <code>/model</code> picker.
<code>ANTHROPIC_DEFAULT_SONNET_MODEL_SUPPORTED_CAPABILITIES</code>	Comma-separated list of capabilities the pinned Sonnet model supports, for example <code>effort,thinking</code> .
<code>ANTHROPIC_FEDERATION_RULE_ID</code>	Federation rule ID for Workload Identity Federation.
<code>ANTHROPIC_FOUNDRY_API_KEY</code>	API key for Microsoft Foundry authentication (see Microsoft Foundry).
<code>ANTHROPIC_FOUNDRY_AUTH_TOKEN</code>	Bearer token for Microsoft Foundry authentication, such as a Microsoft Entra access token.
<code>ANTHROPIC_FOUNDRY_BASE_URL</code>	Full base URL for the Microsoft Foundry resource (for example,...
<code>ANTHROPIC_FOUNDRY_RESOURCE</code>	Microsoft Foundry resource name (for example, <code>my-resource</code>).
<code>ANTHROPIC_MODEL</code>	Name of the model setting to use (see Model Configuration).
<code>ANTHROPIC_ORGANIZATION_ID</code>	Organization ID for Workload Identity Federation.
<code>ANTHROPIC_PROFILE</code>	Name of the Anthropic profile to authenticate with, such as one created by <code>ant auth login</code> or by signing...
<code>ANTHROPIC_SMALL_FAST_MODEL</code>	[DEPRECATED] Name of Haiku-class model for background tasks.
<code>ANTHROPIC_SMALL_FAST_MODEL_AWS_REGION</code>	Override AWS region for the Haiku-class model when using Amazon Bedrock or Amazon Bedrock Mantle.
<code>ANTHROPIC_VERTEX_BASE_URL</code>	Override Google Cloud's Agent Platform endpoint URL. Use for custom Google Cloud's Agent Platform...
<code>ANTHROPIC_VERTEX_PROJECT_ID</code>	GCP project ID that Google Cloud's Agent Platform requests are addressed to.
<code>ANTHROPIC_WORKSPACE_ID</code>	Workspace ID for workload identity federation.
<code>API_FORCE_IDLE_TIMEOUT</code>	Override the 5-minute body idle timeout that aborts a streaming model response when no bytes arrive.
<code>API_TIMEOUT_MS</code>	Timeout for API requests in milliseconds (default: 600000, or 10 minutes; maximum: 2147483647).
<code>AWS_BEARER_TOKEN_BEDROCK</code>	Amazon Bedrock API key for authentication (see Amazon Bedrock API keys).
<code>BASH_DEFAULT_TIMEOUT_MS</code>	Default timeout for a foreground Bash or PowerShell tool command, in milliseconds (default: 120000, or 2...
<code>BASH_MAX_OUTPUT_LENGTH</code>	Maximum number of characters of bash output that Claude Code reads back into a command's result (default:...

<code>BASH_MAX_TIMEOUT_MS</code>	Maximum timeout the model can set for a foreground Bash or PowerShell tool command, in milliseconds...
<code>BETA_TRACING_ENDPOINT</code>	OTLP/HTTP endpoint for detailed beta tracing: with <code>ENABLE_BETA_TRACING_DETAILED=1</code> , logs and traces go...
<code>CCR_FORCE_BUNDLE</code>	Set to <code>1</code> to force <code>claude --cloud</code> to bundle and upload your local repository instead of cloning from...
<code>CLAUDECODE</code>	Set to <code>1</code> in subprocesses Claude Code spawns (Bash and PowerShell tools, tmux sessions, hook commands,...
<code>CLAUDE_AFK_COUNTDOWN_MS</code>	How many milliseconds before auto-continue the on-screen countdown appears on an unanswered...
<code>CLAUDE_AFK_TIMEOUT_MS</code>	How many milliseconds of idle time before an unanswered <code>AskUserQuestion</code> dialog auto-continues without you.
<code>CLAUDE_AGENT_SDK_DISABLE_BUILTIN_AGENTS</code>	Set to <code>1</code> to disable all built-in subagent types such as Explore and Plan.
<code>CLAUDE_AGENT_SDK_MCP_NO_PREFIX</code>	Set to <code>1</code> to skip the <code>mcp_<server>__</code> prefix on tool names from SDK-created MCP servers.
<code>CLAUDE_ASYNC_AGENT_STALL_TIMEOUT_MS</code>	Stall timeout in milliseconds for subagents.
<code>CLAUDE_AUTOCOMPACT_PCT_OVERRIDE</code>	Set the percentage (1-100) of the auto-compact window at which auto-compaction triggers.
<code>CLAUDE_AUTO_BACKGROUND_TASKS</code>	Set to <code>1</code> to force-enable automatic backgrounding of long-running agent tasks.
<code>CLAUDE_AX_PREPARK_MS</code>	In screen reader mode, how many milliseconds Claude Code waits before it writes a new or changed line.
<code>CLAUDE_AX_SCREEN_READER</code>	Set to <code>1</code> to render screen-reader friendly output: flat text without decorative borders or animations.
<code>CLAUDE_AX_STARTUP_QUIET_MS</code>	In screen reader mode, how many milliseconds Claude Code holds the first interface render after the...
<code>CLAUDE_BASH_MAINTAIN_PROJECT_WORKING_DIR</code>	Return to the original working directory after each Bash or PowerShell command in the main session.
<code>CLAUDE_BYTE_STREAM_IDLE_TIMEOUT_MS</code>	Timeout in milliseconds for the byte-level streaming idle watchdog; when set, it takes precedence over...
<code>CLAUDE_CLIENT_PRESENCE_FILE</code>	Path to a file that an external tool, such as a screen-lock listener, creates when you unlock your screen...
<code>CLAUDE_CODE_ACCESSIBILITY</code>	Set to <code>1</code> to keep the native terminal cursor visible and disable the inverted-text cursor indicator.
<code>CLAUDE_CODE_ADDITIONAL_DIRECTORIES_CLAUDE_MD</code>	Set to <code>1</code> to load memory files from directories specified with <code>--add-dir</code> .
<code>CLAUDE_CODE_ALT_SCREEN_FULL_REPAINT</code>	Set to <code>1</code> to repaint the entire screen on every frame in fullscreen rendering instead of sending...
<code>CLAUDE_CODE_ALWAYS_ENABLE_EFFORT</code>	Set to <code>1</code> to send the effort parameter with every request, even when Claude Code does not recognize the...
<code>CLAUDE_CODE_API_KEY_HELPER_TTL_MS</code>	Interval in milliseconds at which credentials should be refreshed (when using <code>apiKeyHelper</code>).
<code>CLAUDE_CODE_ARTIFACT_AUTO_OPEN</code>	Set to <code>0</code> to stop Claude Code from opening the browser automatically when a new artifact is published.

<code>CLAUDE_CODE_ARTIFACT_COMMENTS</code>	Set to <code>0</code> to stop Claude reading and replying to comments on an artifact.
<code>CLAUDE_CODE_ARTIFACT_COMMENTS_AUTOREACT</code>	Set to <code>0</code> to stop Claude replying on its own to comments sent to it.
<code>CLAUDE_CODE_ARTIFACT_PUBLISH_EDITS</code>	Set to <code>0</code> to stop Claude from applying small changes to an artifact's file as part of publishing it.
<code>CLAUDE_CODE_ATTRIBUTION_HEADER</code>	Set to <code>0</code> to omit the attribution block, which carries the client version and a prompt fingerprint, from...
<code>CLAUDE_CODE_AUTO_BACKGROUND_WORKER_CHECKIN_SECONDS</code>	Removed in v2.1.283. Use <code>CLAUDE_CODE_WORKER_CHECKIN_SCHEDULE</code> instead.
<code>CLAUDE_CODE_AUTO_COMPACT_WINDOW</code>	Set the auto-compact window in tokens, from <code>100000</code> to <code>1000000</code> .
<code>CLAUDE_CODE_AUTO_CONNECT_IDE</code>	Override automatic IDE connection.
<code>CLAUDE_CODE_AUTO_MODE_SERVER</code>	Controls whether Claude Code asks the server to review auto mode actions.
<code>CLAUDE_CODE_AWS_CHAIN_RESOLVE_TIMEOUT_MS</code>	Time in milliseconds Claude Code waits for the AWS default credential provider chain to produce...
<code>CLAUDE_CODE_BASH_EDIT_DIFF</code>	Set to <code>0</code> to turn off the diff of the files that changed while a Bash command ran, or <code>1</code> to record it in...
<code>CLAUDE_CODE_BG_TASKS_REPORT_RUNNING</code>	Set to <code>0</code> to make a non-interactive session report an idle status to its host at every turn end, even...
<code>CLAUDE_CODE_BRIDGE_SESSION_ID</code>	Set automatically in Bash tool and hook command subprocesses while the session has an active Remote...
<code>CLAUDE_CODE_BS_AS_CTRL_BACKSPACE</code>	Set to <code>0</code> to make Claude Code read the <code>0x08</code> byte, also written <code>^H</code> , as plain Backspace, or <code>1</code> to read...
<code>CLAUDE_CODE_CERT_STORE</code>	Comma-separated list of CA certificate sources for TLS connections. <code>bundled</code> is the Mozilla CA set...
<code>CLAUDE_CODE_CHILD_SESSION</code>	Set to <code>1</code> in subprocesses Claude Code spawns via the Bash, PowerShell, and Monitor tools, hook commands,...
<code>CLAUDE_CODE_CLIENT_CERT</code>	Path to client certificate file for mTLS authentication.
<code>CLAUDE_CODE_CLIENT_KEY</code>	Path to client private key file for mTLS authentication.
<code>CLAUDE_CODE_CLIENT_KEY_PASSPHRASE</code>	Passphrase for encrypted <code>CLAUDE_CODE_CLIENT_KEY</code> (optional).
<code>CLAUDE_CODE_CONNECT_TIMEOUT_MS</code>	Removed in v2.1.186 and now a no-op.
<code>CLAUDE_CODE_DEBUG_LOGS_DIR</code>	Override the debug log file path.
<code>CLAUDE_CODE_DEBUG_LOG_LEVEL</code>	Minimum log level written to the debug log file.
<code>CLAUDE_CODE_DISABLE_1M_CONTEXT</code>	Set to <code>1</code> to turn off 1M context window support.
<code>CLAUDE_CODE_DISABLE_ADAPTIVE_THINKING</code>	Set to <code>1</code> to disable adaptive reasoning on Opus 4.6 and Sonnet 4.6 and fall back to the fixed thinking...
<code>CLAUDE_CODE_DISABLE_ADMIN_ENV_UNION</code>	Set to <code>1</code> to stop Claude Code from merging managed settings <code>env</code> blocks per key across admin sources, so...
<code>CLAUDE_CODE_DISABLE_ADVISOR_TOOL</code>	Set to <code>1</code> to disable the advisor tool.

<code>CLAUDE_CODE_DISABLE_AGENT_VIEW</code>	Set to <code>1</code> to turn off background agents and agent view: <code>claude agents</code> , <code>--bg</code> , <code>/background</code> , and the...
<code>CLAUDE_CODE_DISABLE_ALTERNATE_SCREEN</code>	Set to <code>1</code> to disable fullscreen rendering and use the classic main-screen renderer.
<code>CLAUDE_CODE_DISABLE_ARTIFACT</code>	Set to <code>1</code> to turn off the Artifact tool, which publishes session output as a private web page on claude.ai.
<code>CLAUDE_CODE_DISABLE_ATTACHMENTS</code>	Set to <code>1</code> to disable attachment processing.
<code>CLAUDE_CODE_DISABLE_AUTH_REFRESH_LOCK</code>	Set to <code>1</code> to make a Claude Code process run its <code>gcpAuthRefresh</code> or <code>awsAuthRefresh</code> command itself...
<code>CLAUDE_CODE_DISABLE_AUTO_MEMORY</code>	Set to <code>1</code> to disable auto memory.
<code>CLAUDE_CODE_DISABLE_BACKGROUND_TASKS</code>	Set to <code>1</code> to disable all background task functionality, including the <code>run_in_background</code> parameter on...
<code>CLAUDE_CODE_DISABLE_BEDROCK_CONTENT_TYPE_DEFAULT</code>	Set to <code>1</code> to stop Claude Code from treating an Amazon Bedrock streaming response with a missing or empty...
<code>CLAUDE_CODE_DISABLE_BEDROCK_CONTENT_TYPE_GUARD</code>	Set to <code>1</code> to skip the check that an Amazon Bedrock streaming response carries the...
<code>CLAUDE_CODE_DISABLE_BG_EXIT_HANDOFF</code>	Set to <code>1</code> to stop a background session's running background shell commands, dynamic workflows, and, as of...
<code>CLAUDE_CODE_DISABLE_BG_SHELL_PRESSURE_REAP</code>	Set to <code>1</code> to stop Claude Code from terminating background shell commands under memory pressure.
<code>CLAUDE_CODE_DISABLE_BUNDLED_SKILLS</code>	Set to <code>1</code> to disable the skills and workflows included with Claude Code: bundled skills and workflows are...
<code>CLAUDE_CODE_DISABLE_CFC_PROMPT</code>	Set to <code>1</code> to keep the Claude in Chrome browser tools available while omitting the Chrome section of the...
<code>CLAUDE_CODE_DISABLE_CLAUDE_MDS</code>	Set to <code>1</code> to prevent loading any CLAUDE.md memory files into context, including user, project, and auto...
<code>CLAUDE_CODE_DISABLE_CRON</code>	Set to <code>1</code> to disable scheduled tasks.
<code>CLAUDE_CODE_DISABLE_DANGEROUS_RM_TIMEOUT</code>	Set to <code>1</code> to turn off the time limit on critical-path removal prompts.
<code>CLAUDE_CODE_DISABLE_EXPERIMENTAL_BETAS</code>	Set to <code>1</code> to strip pre-release <code>anthropic-beta</code> request headers, the body fields that pair with them, and...
<code>CLAUDE_CODE_DISABLE_EXPLORE_PLAN_AGENTS</code>	Set to <code>1</code> to disable the built-in Explore and Plan subagents.
<code>CLAUDE_CODE_DISABLE_FAST_MODE</code>	Set to <code>1</code> to disable fast mode.
<code>CLAUDE_CODE_DISABLE_FEEDBACK_SURVEY</code>	Set to <code>1</code> to disable the "How is Claude doing?" session quality surveys.
<code>CLAUDE_CODE_DISABLE_FILE_CHECKPOINTING</code>	Set to <code>1</code> to disable file checkpointing.
<code>CLAUDE_CODE_DISABLE_FILE_MENTIONS</code>	Set to <code>1</code> to stop Claude Code from expanding <code>@</code> mentions of files and directories.
<code>CLAUDE_CODE_DISABLE_GIT_INSTRUCTIONS</code>	Set to <code>1</code> to remove built-in commit and PR workflow instructions and the git status snapshot from...
<code>CLAUDE_CODE_DISABLE_INLINE_SHELL_RM_PROMPT</code>	Set to <code>1</code> to stop Claude Code from reading scripts passed to a shell with <code>-c</code> , such as <code>'bash -c 'rm -rf...</code>
<code>CLAUDE_CODE_DISABLE_LEGACY_MODEL_REMAP</code>	Set to <code>1</code> to prevent automatic remapping of Opus 4 and Opus 4.1 to the current Opus version on the...

<code>CLAUDE_CODE_DISABLE_MODEL_ACCESS_FALLBACK</code>	Set to <code>1</code> to stop Claude Code on Amazon Bedrock and Google Cloud's Agent Platform from switching to an...
<code>CLAUDE_CODE_DISABLE_MOUSE</code>	Set to <code>1</code> to disable mouse tracking in fullscreen rendering.
<code>CLAUDE_CODE_DISABLE_MOUSE_CLICKS</code>	Set to <code>1</code> to disable click, drag, and hover handling in fullscreen rendering while keeping mouse-wheel...
<code>CLAUDE_CODE_DISABLE_MTLS_RELOAD_ON_STALE_CONNECTION</code>	Set to <code>1</code> to stop Claude Code from re-reading the mTLS client certificate and key when an API request...
<code>CLAUDE_CODE_DISABLE_NONESSENTIAL_TRAFFIC</code>	Set to any non-empty value, such as <code>1</code> , to disable nonessential network traffic: auto-updates, telemetry,...
<code>CLAUDE_CODE_DISABLE_NONSTREAMING_FALLBACK</code>	Set to <code>1</code> to disable the non-streaming fallback when a streaming request fails mid-stream.
<code>CLAUDE_CODE_DISABLE_NOTIFICATION_PRESENCE_CHECK</code>	Set to <code>1</code> to send the <code>PushNotification</code> tool's desktop notification even while you are typing in or...
<code>CLAUDE_CODE_DISABLE_OFFICIAL_MARKETPLACE_AUTOINSTALL</code>	Set to <code>1</code> to disable automatic registration of the official plugin marketplace.
<code>CLAUDE_CODE_DISABLE_PERMISSION_PROMPT_NOTIFY_HOOKS</code>	Set to <code>1</code> to stop Claude Code from running your <code>Notification</code> hooks for unanswered permission requests...
<code>CLAUDE_CODE_DISABLE_POLICY_SKILLS</code>	Set to <code>1</code> to skip loading skills from the system-wide managed skills directory.
<code>CLAUDE_CODE_DISABLE_POWERSHELL_CMD_RM_DENY</code>	Set to <code>1</code> to turn off the PowerShell tool check that denies the <code>cmd</code> built-ins <code>rd</code> , <code>rmdir</code> , <code>del</code> , and...
<code>CLAUDE_CODE_DISABLE_REFUSAL_FALLBACK</code>	Set to <code>1</code> to turn off the automatic model switch when a safety classifier flags a request, the behavior...
<code>CLAUDE_CODE_DISABLE_STRUCTURED_OUTPUTS</code>	Set to <code>1</code> to stop Claude Code from sending the structured-output <code>output_config.format</code> field and the...
<code>CLAUDE_CODE_DISABLE_SUBSTITUTION_RM_PROMPT</code>	Set to <code>1</code> to turn off the critical-path check for a recursive <code>rm</code> whose target is entirely the output of...
<code>CLAUDE_CODE_DISABLE_TERMINAL_TITLE</code>	Set to <code>1</code> to disable automatic terminal title updates based on conversation context.
<code>CLAUDE_CODE_DISABLE_THINKING</code>	Set to <code>1</code> to omit the <code>thinking</code> parameter from API requests entirely.
<code>CLAUDE_CODE_DISABLE_UNKNOWN_MODEL_WINDOW_ENFORCEMENT</code>	Set to <code>1</code> to skip proactive auto-compaction when Claude Code doesn't recognize the model ID, such as an...
<code>CLAUDE_CODE_DISABLE_VIRTUAL_SCROLL</code>	Set to <code>1</code> to disable virtual scrolling in fullscreen rendering and render every message in the transcript.
<code>CLAUDE_CODE_DISABLE_WEB_FETCH</code>	Set to <code>1</code> to turn off the WebFetch tool.
<code>CLAUDE_CODE_DISABLE_WINDOWS_SHELL_LAUNCHER</code>	Set to <code>1</code> to start PowerShell tool commands on Windows directly instead of through the <code>cmd.exe</code> launcher.
<code>CLAUDE_CODE_DISABLE_WORKFLOWS</code>	Set to <code>1</code> to disable workflows.
<code>CLAUDE_CODE_EFFORT_LEVEL</code>	Set the effort level for supported models.
<code>CLAUDE_CODE_EMIT_SESSION_STATE_EVENTS</code>	Set to <code>1</code> to add <code>session_state_changed</code> messages, which carry the session's state, to the message stream.
<code>CLAUDE_CODE_ENABLE_AUTO_MODE</code>	Accepted for compatibility with older releases and has no effect.

<code>CLAUDE_CODE_ENABLE_AWAY_SUMMARY</code>	Override session recap availability.
<code>CLAUDE_CODE_ENABLE_BACKGROUND_PLUGIN_REFRESH</code>	Set to <code>1</code> to refresh plugin state at turn boundaries in non-interactive mode after a background install...
<code>CLAUDE_CODE_ENABLE_CFC</code>	Set to <code>1</code> to start a CLI session with Chrome integration on, or <code>0</code> to start with it off.
<code>CLAUDE_CODE_ENABLE_FEEDBACK_SURVEY_FOR_OTEL</code>	Set to <code>1</code> to route the “How is Claude doing?” session quality survey to your own OpenTelemetry collector...
<code>CLAUDE_CODE_ENABLE_FINE_GRAINED_TOOL_STREAMING</code>	Controls whether tool call inputs stream from the API as Claude generates them.
<code>CLAUDE_CODE_ENABLE_GATEWAY_MODEL_DISCOVERY</code>	Set to <code>1</code> to populate the <code>/model</code> picker from your gateway's <code>/v1/models</code> endpoint when...
<code>CLAUDE_CODE_ENABLE_OPUS_4_7_FAST_MODE</code>	Removed in v2.1.142, when the fast mode default moved from Opus 4.6 to Opus 4.7.
<code>CLAUDE_CODE_ENABLE_PROMPT_SUGGESTION</code>	Set to <code>false</code> to turn off prompt suggestions, the grayed-out predictions that appear in your prompt input.
<code>CLAUDE_CODE_ENABLE_TASKS</code>	Selects which task-tracking tools Claude Code provides in sessions that have them.
<code>CLAUDE_CODE_ENABLE_TELEMETRY</code>	Set to <code>1</code> to enable OpenTelemetry data collection for metrics and logging.
<code>CLAUDE_CODE_ENABLE_TODO_TOOLS</code>	Set to <code>1</code> to get the task-tracking tools on every model.
<code>CLAUDE_CODE_EXIT_AFTER_STOP_DELAY</code>	Time in milliseconds to wait after the query loop becomes idle before automatically exiting.
<code>CLAUDE_CODE_EXPERIMENTAL_AGENT_TEAMS</code>	Set to <code>1</code> to enable agent teams.
<code>CLAUDE_CODE_EXTRA_BODY</code>	JSON object to merge into the top level of every API request body.
<code>CLAUDE_CODE_FILE_READ_MAX_OUTPUT_TOKENS</code>	Override the default token limit for file reads, which is 25,000 tokens.
<code>CLAUDE_CODE_FORCE_SESSION_PERSISTENCE</code>	Set to <code>1</code> to force transcript persistence, prompt history, and <code>claude agents</code> registration even when...
<code>CLAUDE_CODE_FORCE_STRIKETHROUGH</code>	Set to <code>1</code> to force strikethrough rendering for <code>~~text~~</code> in Claude's responses when your terminal...
<code>CLAUDE_CODE_FORCE_SYNC_OUTPUT</code>	Set to <code>1</code> to force-enable DEC private mode 2026 synchronized output when your terminal supports it but is...
<code>CLAUDE_CODE_FORCE_TERMINAL_IMAGES</code>	Set to <code>1</code> to draw mod <code>Image</code> elements as pictures when your terminal draws kitty graphics protocol...
<code>CLAUDE_CODE_FORK_SUBAGENT</code>	Controls fork mode, which lets Claude spawn forked subagents itself and is on by default in interactive...
<code>CLAUDE_CODE_FORWARD_SUBAGENT_TEXT</code>	Set to <code>1</code> to emit subagent text and thinking blocks in <code>claude -p -output-format stream-json</code> output...
<code>CLAUDE_CODE_GATEWAY_HINT_HEADERS</code>	Set to <code>1</code> to send the gateway hint headers, such as <code>x-claude-code-request-class</code> and...
<code>CLAUDE_CODE_GATEWAY_MODEL_DISCOVERY_TIMEOUT_MS</code>	Timeout in milliseconds for the gateway model discovery request that...
<code>CLAUDE_CODE_GIT_BASH_PATH</code>	Windows only: path to the Git Bash executable (<code>bash.exe</code>).

<code>CLAUDE_CODE_GLOB_HIDDEN</code>	Set to <code>false</code> to exclude dotfiles from results when Claude invokes the Glob tool.
<code>CLAUDE_CODE_GLOB_NO_IGNORE</code>	Set to <code>false</code> to make the Glob tool respect <code>.gitignore</code> patterns.
<code>CLAUDE_CODE_GLOB_TIMEOUT_SECONDS</code>	Timeout in seconds for Glob tool file discovery.
<code>CLAUDE_CODE_GOAL_CHECKIN_MINUTES</code>	How many minutes background work can keep an active goal waiting before Claude Code asks Claude to check...
<code>CLAUDE_CODE_GZIP_REQUEST_BODIES</code>	Set to <code>0</code> to turn off gzip compression of Claude API, telemetry, and artifact publish request bodies sent...
<code>CLAUDE_CODE_HIDE_CWD</code>	Set to <code>1</code> to hide the working directory in the startup logo.
<code>CLAUDE_CODE_IDE_HOST_OVERRIDE</code>	Override the host address used to connect to the IDE extension.
<code>CLAUDE_CODE_IDE_SKIP_AUTO_INSTALL</code>	Set to <code>1</code> to skip auto-installation of IDE extensions.
<code>CLAUDE_CODE_IDE_SKIP_VALID_CHECK</code>	Set to <code>1</code> to skip validation of IDE lockfile entries during connection.
<code>CLAUDE_CODE_IDLE_COMPACT_MIN_TOKENS</code>	Context size, in tokens, at which idle compaction applies to a conversation (default: <code>200000</code>).
<code>CLAUDE_CODE_MAX_CONCURRENT_SUBAGENTS</code>	How many subagents can be running in one session before the Agent tool refuses to spawn another (default: <code>20</code>).
<code>CLAUDE_CODE_MAX_CONTEXT_TOKENS</code>	Override the context window size Claude Code assumes for the active model.
<code>CLAUDE_CODE_MAX_MCP_DESCRIPTION_LENGTH</code>	Maximum length in characters of MCP tool descriptions and MCP server instructions that Claude Code sends...
<code>CLAUDE_CODE_MAX_OUTPUT_TOKENS</code>	Set the maximum number of output tokens for most requests.
<code>CLAUDE_CODE_MAX_RETRIES</code>	Override the number of times to retry failed API requests (default: <code>10</code>).
<code>CLAUDE_CODE_MAX_SUBAGENTS_PER_SESSION</code>	Removed in v2.1.224 and now a no-op.
<code>CLAUDE_CODE_MAX_SUBAGENT_SPAWN_DEPTH</code>	Number of subagent layers allowed below the main conversation (default: <code>3</code>).
<code>CLAUDE_CODE_MAX_TOOL_USE_CONCURRENCY</code>	Maximum number of read-only tools and subagents that can execute in parallel (default: <code>10</code>).
<code>CLAUDE_CODE_MAX_TURNS</code>	Cap the number of agentic turns when no explicit limit is passed.
<code>CLAUDE_CODE_MAX_WEB_SEARCHES_PER_SESSION</code>	Cap on WebSearch calls (default: <code>200</code>).
<code>CLAUDE_CODE_MCP_ALLOWLIST_ENV</code>	Set to <code>1</code> to spawn stdio MCP servers with only a safe baseline environment plus the server's configured...
<code>CLAUDE_CODE_MCP_AUTO_BACKGROUND_MS</code>	Elapsed time in milliseconds before a still-running MCP tool call moves to a background task (default:...
<code>CLAUDE_CODE_MCP_STARTUP_WAIT_MS</code>	How long in milliseconds the first turn of a non-interactive session waits for MCP servers that are still...
<code>CLAUDE_CODE_MCP_TOOL_IDLE_TIMEOUT</code>	Idle timeout in milliseconds for MCP tool calls.
<code>CLAUDE_CODE_MESSAGING_SOCKET</code>	Set by Claude Code, not by you: in sessions that bind an inbox socket, Claude Code exports that socket's...
<code>CLAUDE_CODE_MESSAGING_TOKEN</code>	Set by Claude Code, not by you: in sessions that bind an inbox socket, Claude Code exports this...
<code>CLAUDE_CODE_NATIVE_CURSOR</code>	Set to <code>1</code> to show the terminal's own cursor at the input caret instead of a drawn block.

<code>CLAUDE_CODE_NEW_INIT</code>	Set to <code>1</code> to make <code>/init</code> run an interactive setup flow.
<code>CLAUDE_CODE_NONBLOCKING_STDOUT</code>	Set to <code>1</code> to write terminal output through a second non-blocking file descriptor, so a terminal that...
<code>CLAUDE_CODE_NONSTREAMING_TIMEOUT_RETRIES</code>	Limit how many times Claude Code re-sends a non-streaming request that times out.
<code>CLAUDE_CODE_NO_FLICKER</code>	Set to <code>1</code> to enable fullscreen rendering, a research preview that reduces flicker and keeps memory flat...
<code>CLAUDE_CODE_OAUTH_REFRESH_TOKEN</code>	OAuth refresh token for Claude.ai authentication.
<code>CLAUDE_CODE_OAUTH_SCOPES</code>	Space-separated OAuth scopes the refresh token was issued with, such as <code>"user:profile user:inference..."</code>
<code>CLAUDE_CODE_OAUTH_TOKEN</code>	OAuth access token for claude.ai authentication.
<code>CLAUDE_CODE_OPUS_4_6_FAST_MODE_OVERRIDE</code>	Removed in v2.1.160 and now a no-op.
<code>CLAUDE_CODE_OTEL_CONTENT_MAX_LENGTH</code>	Maximum length of content-bearing OpenTelemetry attributes (model responses, tool content, system prompts,...)
<code>CLAUDE_CODE_OTEL_DIAG_STDERR</code>	Set to <code>1</code> to write OpenTelemetry exporter diagnostic errors to stderr.
<code>CLAUDE_CODE_OTEL_FLUSH_TIMEOUT_MS</code>	Timeout in milliseconds for flushing pending OpenTelemetry spans (default: 5000).
<code>CLAUDE_CODE_OTEL_HEADERS_HELPER_DEBOUNCE_MS</code>	Interval for refreshing dynamic OpenTelemetry headers in milliseconds (default: 1740000 / 29 minutes).
<code>CLAUDE_CODE_OTEL_SHUTDOWN_TIMEOUT_MS</code>	Timeout in milliseconds for the OpenTelemetry exporter to finish on shutdown (default: 2000).
<code>CLAUDE_CODE_OVERLOADED_RETRY_BASE_DELAY_MS</code>	Starting delay in milliseconds, in place of the default 500, of the exponential backoff between automatic...
<code>CLAUDE_CODE_OVERLOADED_RETRY_MAX_DELAY_MS</code>	Ceiling in milliseconds, in place of the default 32000, on the exponential backoff between automatic...
<code>CLAUDE_CODE_PACKAGE_MANAGER_AUTO_UPDATE</code>	Set to <code>1</code> to let Claude Code run your package manager's upgrade command in the background when a new...
<code>CLAUDE_CODE_PERFORCE_MODE</code>	Set to <code>1</code> to enable Perforce-aware write protection.
<code>CLAUDE_CODE_PLUGIN_CACHE_DIR</code>	Override the plugins root directory.
<code>CLAUDE_CODE_PLUGIN_DIRS</code>	Plugin directories to load for the session, each loaded the way a <code>-plugin-dir</code> flag loads it.
<code>CLAUDE_CODE_PLUGIN_DIR_WATCH</code>	Controls whether Claude Code reloads a mod when the mod's files change.
<code>CLAUDE_CODE_PLUGIN_GIT_TIMEOUT_MS</code>	Timeout in milliseconds for cloning or refreshing a plugin marketplace (default: 120000).
<code>CLAUDE_CODE_PLUGIN_KEEP_MARKETPLACE_ON_FAILURE</code>	Set to <code>1</code> to skip the re-clone attempt and keep using the existing marketplace checkout when a...
<code>CLAUDE_CODE_PLUGIN_PREFER_HTTPS</code>	Set to <code>1</code> to clone GitHub <code>owner/repo</code> shorthand sources over HTTPS instead of SSH. Applies to plugin...
<code>CLAUDE_CODE_PLUGIN_SEED_DIR</code>	Path to one or more read-only plugin seed directories, separated by <code>:</code> on Unix or <code>;</code> on Windows.
<code>CLAUDE_CODE_POWERSHELL_RESPECT_EXECUTION_POLICY</code>	Set to <code>1</code> to stop Claude Code from passing <code>-ExecutionPolicy Bypass</code> when spawning PowerShell for tool...

<code>CLAUDE_CODE_PRINT_BG_WAIT_CEILING_MS</code>	Ceiling in milliseconds on idle waiting for background work, such as subagents and workflows, after the...
<code>CLAUDE_CODE_PROCESS_WRAPPER</code>	Launch the processes Claude Code starts from its own binary, such as the background service that hosts...
<code>CLAUDE_CODE_PROJECT_DIR_NAME</code>	Set together with <code>CLAUDE_CONFIG_DIR</code> to choose the <code>projects/</code> directory name Claude Code stores that...
<code>CLAUDE_CODE_PROMPT_CACHE_TTL</code>	Set <code>5m</code> or <code>1h</code> , the only values Claude Code accepts, to choose the prompt cache TTL for the main...
<code>CLAUDE_CODE_PROPAGATE_TRACEPARENT</code>	Set to <code>1</code> to propagate W3C trace context when <code>ANTHROPIC_BASE_URL</code> points at a custom proxy.
<code>CLAUDE_CODE_PROVIDER_MANAGED_BY_HOST</code>	Set by host platforms that embed Claude Code and manage model provider routing on its behalf.
<code>CLAUDE_CODE_PROXY_RESOLVES_HOSTS</code>	Set to <code>1</code> to allow the proxy to perform DNS resolution instead of the caller.
<code>CLAUDE_CODE_REMOTE</code>	Set automatically to <code>true</code> when Claude Code is running as a cloud session.
<code>CLAUDE_CODE_REMOTE_SESSION_ID</code>	Set automatically in cloud sessions to the current session's ID. Read this to construct a link back to the...
<code>CLAUDE_CODE_RESTRICTED</code>	Set to <code>1</code> to start the session in restricted mode, the same as passing <code>--restricted</code> .
<code>CLAUDE_CODE_RESUME_INTERRUPTED_TURN</code>	Set to <code>1</code> to automatically resume if the previous session ended mid-turn.
<code>CLAUDE_CODE_RESUME_INTERRUPTED_TURN_MAX_AGE_MS</code>	Maximum age in milliseconds of the last transcript message for a session that ended mid-turn to continue...
<code>CLAUDE_CODE_RESUME_PROMPT</code>	Override the continuation message Claude Code sends to Claude when <code>CLAUDE_CODE_RESUME_INTERRUPTED_TURN</code> ...
<code>CLAUDE_CODE_RETRY_WATCHDOG</code>	Set to <code>1</code> for unattended sessions such as eval harnesses, CI jobs, or remote workers.
<code>CLAUDE_CODE_RETRY_WATCHDOG_MAX_WAIT_MS</code>	Maximum time in milliseconds that each API request spends waiting out <code>429</code> and <code>529</code> errors when...
<code>CLAUDE_CODE_SAFE_MODE</code>	Set to <code>1</code> to start in safe mode: CLAUDE.md, skills, plugins, hooks, MCP servers, custom commands and...
<code>CLAUDE_CODE_SCRIPT_CAPS</code>	JSON object limiting how many times specific scripts may be invoked per session when...
<code>CLAUDE_CODE_SCROLL_SPEED</code>	Set the mouse wheel scroll multiplier in fullscreen rendering.
<code>CLAUDE_CODE_SEND_FEEDBACK</code>	Set to <code>0</code> to turn off Claude-drafted feedback for a session.
<code>CLAUDE_CODE_SESSIONEND_HOOKS_TIMEOUT_MS</code>	Override the time budget in milliseconds for SessionEnd hooks.
<code>CLAUDE_CODE_SESSION_ID</code>	Set automatically to the current session ID in Bash and PowerShell tool subprocesses, hook command...
<code>CLAUDE_CODE_SHELL</code>	Set the shell Claude Code uses to run Bash tool commands.
<code>CLAUDE_CODE_SHELL_PREFIX</code>	Command prefix that wraps shell commands Claude Code spawns: Bash tool calls, hook commands, status line...
<code>CLAUDE_CODE_SIMPLE</code>	Set to <code>1</code> to run with a minimal system prompt and only the Bash, file read, and file edit tools.

<code>CLAUDE_CODE_SIMPLE_SYSTEM_PROMPT</code>	Choose between Claude Code's full system prompt and a shorter one with abbreviated tool descriptions.
<code>CLAUDE_CODE_SKIP_ANTHROPIC_AWS_AUTH</code>	Skip client-side authentication for Claude Platform on AWS, for gateways that sign requests themselves.
<code>CLAUDE_CODE_SKIP_AWS_CRED_CACHE</code>	Set to <code>1</code> to turn off the in-process cache of credentials resolved from the AWS default credential...
<code>CLAUDE_CODE_SKIP_BEDROCK_AUTH</code>	Skip AWS authentication for Amazon Bedrock (for example, when using an LLM gateway).
<code>CLAUDE_CODE_SKIP_FAST_MODE_NETWORK_ERRORS</code>	Set to <code>1</code> to treat a failed fast mode availability check as available, for networks that block the...
<code>CLAUDE_CODE_SKIP_FAST_MODE_ORG_CHECK</code>	Set to <code>1</code> to skip the client-side fast mode availability check, for proxies that intercept the check's...
<code>CLAUDE_CODE_SKIP_FOUNDRY_AUTH</code>	Skip Azure authentication for Microsoft Foundry, for a proxy or gateway that injects its own...
<code>CLAUDE_CODE_SKIP_MANTLE_AUTH</code>	Skip AWS authentication for Amazon Bedrock Mantle (for example, when using an LLM gateway).
<code>CLAUDE_CODE_SKIP_MODEL_ACCESS_MEMORY</code>	The startup model checks on Amazon Bedrock and Google Cloud's Agent Platform remember on this machine...
<code>CLAUDE_CODE_SKIP_PROMPT_HISTORY</code>	Set to <code>1</code> to skip writing prompt history and session transcripts to disk.
<code>CLAUDE_CODE_SKIP_VERTEX_AUTH</code>	Skip Google authentication for Google Cloud's Agent Platform (for example, when using an LLM gateway).
<code>CLAUDE_CODE_STARTUP_FAILURE_RESULTS</code>	Set to <code>1</code> to have a session started with <code>--output-format stream-json</code> write a result message naming why...
<code>CLAUDE_CODE_STOP_HOOK_BLOCK_CAP</code>	Maximum number of consecutive times a Stop or SubagentStop hook may block the turn from ending before...
<code>CLAUDE_CODE_SUBAGENT_MODEL</code>	The default model for subagents, agent team teammates, and workflow agents that aren't assigned a model...
<code>CLAUDE_CODE_SUBAGENT_MODEL_FORCE</code>	Set to <code>1</code> to force one model onto subagents, teammates, and workflow agents.
<code>CLAUDE_CODE_SUBAGENT_PROMPT_CACHE_TTL</code>	Set <code>5m</code> or <code>1h</code> , the only values Claude Code accepts, to choose the prompt cache TTL for requests outside...
<code>CLAUDE_CODE_SUBPROCESS_ENV_SCRUB</code>	Set to <code>1</code> to strip credentials from the environments of the subprocesses Claude Code starts, such as Bash...
<code>CLAUDE_CODE_SYNC_PLUGIN_INSTALL</code>	Set to <code>1</code> in non-interactive mode (the <code>-p</code> flag) to wait for plugin installation to complete before the...
<code>CLAUDE_CODE_SYNC_PLUGIN_INSTALL_TIMEOUT_MS</code>	Timeout in milliseconds for synchronous plugin installation.
<code>CLAUDE_CODE_SYNC_SKILLS</code>	Set to <code>1</code> in non-interactive mode with the <code>-p</code> flag to make Claude Code download the skills enabled for...
<code>CLAUDE_CODE_SYNC_SKILLS_INSTALL_TIMEOUT_MS</code>	Timeout in milliseconds for the skills resync that runs mid-session when an app built on the Agent SDK...
<code>CLAUDE_CODE_SYNC_SKILLS_WAIT_TIMEOUT_MS</code>	Timeout in milliseconds for the first query to wait for the initial skill list when...
<code>CLAUDE_CODE_SYNTAX_HIGHLIGHT</code>	Set to <code>false</code> to disable syntax highlighting in diff output.
<code>CLAUDE_CODE_TASK_LIST_ID</code>	Share a task list across sessions.

<code>CLAUDE_CODE_TEAM_TEARDOWN_PARK_TIMEOUT_MS</code>	Override, in milliseconds, how long a non-interactive session waits at exit for its agent team to finish...
<code>CLAUDE_CODE_TMPDIR</code>	Override the temp directory used for internal temp files.
<code>CLAUDE_CODE_TMUX_TRUECOLOR</code>	Set to any non-empty value, such as <code>1</code> , to allow 24-bit truecolor output inside tmux. **Setting it to <code>0</code> ...
<code>CLAUDE_CODE_TOOL_MEMORY_CGROUP_EXCLUDE</code>	On Linux and WSL, set to a comma-separated list of the kinds of processes Claude Code excludes from the...
<code>CLAUDE_CODE_TOOL_MEMORY_LIMIT</code>	On Linux and WSL, set to a size such as <code>4G</code> to cap the memory that Bash and PowerShell tool commands can...
<code>CLAUDE_CODE_TRANSCRIPT_LOCAL_GC</code>	Set to <code>1</code> to limit how large the transcript file of a long <code>-p</code> or Agent SDK session grows.
<code>CLAUDE_CODE_USER_DIALOG_TIMEOUT_MS</code>	Deadline in milliseconds before Claude Code cancels a dialog it forwards to a remote client such as a...
<code>CLAUDE_CODE_USE_ANTHROPIC_AWS</code>	Use Claude Platform on AWS.
<code>CLAUDE_CODE_USE_BEDROCK</code>	Use Amazon Bedrock.
<code>CLAUDE_CODE_USE_FOUNDRY</code>	Use Microsoft Foundry.
<code>CLAUDE_CODE_USE_MANTLE</code>	Use the Amazon Bedrock Mantle endpoint.
<code>CLAUDE_CODE_USE_NATIVE_FILE_SEARCH</code>	Set to <code>1</code> to discover custom commands, subagents, and output styles using Node.js file APIs instead of...
<code>CLAUDE_CODE_USE_POWERSHELL_TOOL</code>	Controls the PowerShell tool.
<code>CLAUDE_CODE_USE_VERTEX</code>	Use Google Cloud's Agent Platform.
<code>CLAUDE_CODE_WEBFETCH_CACHE_TTL_MS</code>	Set to the number of milliseconds WebFetch keeps each fetched URL's response cached.
<code>CLAUDE_CODE_WEBFETCH_DEADLINE_MS</code>	Upper bound in milliseconds on how long WebFetch waits for a page to download, including any redirects it...
<code>CLAUDE_CODE_WEB_SEARCH_REFILLS_PER_HOUR</code>	Rate at which a session's WebSearch limit refills, in calls per hour.
<code>CLAUDE_CODE_WORKER_CHECKIN_SCHEDULE</code>	When <code>CLAUDE_AUTO_BACKGROUND_TASKS</code> is set to <code>1</code> , how long Claude Code waits before each reminder to...
<code>CLAUDE_CODE_WORKFLOW_MAX_CONCURRENT_AGENTS</code>	How many agents a single workflow run executes at once, from <code>1</code> to <code>256</code> .
<code>CLAUDE_CODE_WORKFLOW_PREFIX_STAGGER_MS</code>	Upper bound in milliseconds on how long a workflow agent waits for a same-prefix sibling's first response...
<code>CLAUDE_CODE_YOU_SHOULD_KNOW_DEBUG</code>	Set to <code>1</code> to show, above the prompt, a line for each of the last three checks the You should know mod...
<code>CLAUDE_CONFIG_DIR</code>	Override the configuration directory (default: <code>~/ .claude</code>).
<code>CLAUDE_DISABLE_ADOPT</code>	Set to <code>1</code> to stop in-flight background work instead of carrying it over when you background a session by...
<code>CLAUDE Effort</code>	Set automatically in Bash tool subprocesses and hook commands to the effort level in effect when the...
<code>CLAUDE_ENABLE_BYTE_WATCHDOG</code>	Set to <code>1</code> to force-enable the byte-level streaming idle watchdog, or set to <code>0</code> to force-disable it. <code>0</code> ...
<code>CLAUDE_ENABLE_BYTE_WATCHDOG_BEDROCK</code>	Set to <code>1</code> to enable the byte-level streaming idle watchdog on Amazon Bedrock <code>vnd.amazon.eventstream</code> ...

<code>CLAUDE_ENABLE_STREAM_WATCHDOG</code>	Set to <code>0</code> to force-disable the event-level streaming idle watchdog, or set to <code>1</code> to force-enable it.
<code>CLAUDE_ENV_FILE</code>	Path to a shell script whose contents Claude Code runs before each Bash command in the same shell process,...
<code>CLAUDE_JOB_DIR</code>	Set by Claude Code in each background session to that session's <code>~/.claude/jobs/<id></code> directory.
<code>CLAUDE_PID</code>	Claude Code sets this to its own process ID in the subprocesses it spawns: Bash and PowerShell tool...
<code>CLAUDE_REMOTE_CONTROL_SESSION_NAME_PREFIX</code>	Prefix for auto-generated Remote Control session names when no explicit name is provided.
<code>CLAUDE_STREAM_FIRST_BYTE_TIMEOUT_MS</code>	Deadline in milliseconds for the first response byte of a streaming request, on the connections where the...
<code>CLAUDE_STREAM_IDLE_TIMEOUT_MS</code>	Timeout in milliseconds before the event- and byte-level streaming idle watchdogs close a stalled connection.
<code>CLAUDE_SUBAGENT_BG_SHELL_MAX_MS</code>	Removed in v2.1.260 and now a no-op.
<code>DEBUG</code>	Set to <code>1</code> to enable debug mode, equivalent to launching with <code>--debug</code> .
<code>DISABLE_AUTOUPDATER</code>	Set to <code>1</code> to disable automatic background updates.
<code>DISABLE_AUTO_COMPACT</code>	Set to <code>1</code> to disable automatic compaction when approaching the context limit, along with idle compaction.
<code>DISABLE_COMPACT</code>	Set to <code>1</code> to disable all compaction: both automatic compaction and the manual <code>/compact</code> command.
<code>DISABLE_COST_WARNINGS</code>	Set to <code>1</code> to disable cost warning messages.
<code>DISABLE_DOCTOR_COMMAND</code>	Set to <code>1</code> to hide the <code>/doctor</code> setup checkup skill and its <code>/checkup</code> alias.
<code>DISABLE_ERROR_REPORTING</code>	Set to any non-empty value, such as <code>1</code> , to opt out of error reporting. **Setting it to <code>0</code> or <code>false</code> ...
<code>DISABLE_EXTRA_USAGE_COMMAND</code>	Set to <code>1</code> to hide the <code>/usage-credits</code> command that lets users purchase additional usage beyond rate limits.
<code>DISABLE_FEEDBACK_COMMAND</code>	Set to <code>1</code> to disable the <code>/feedback</code> command and Claude-drafted feedback.
<code>DISABLE_GROWTHBOOK</code>	Set to <code>1</code> or <code>true</code> to disable GrowthBook feature-flag fetching and use code defaults for every flag.
<code>DISABLE_INSTALLATION_CHECKS</code>	Set to <code>1</code> to disable installation warnings.
<code>DISABLE_INSTALL_GITHUB_APP_COMMAND</code>	Set to <code>1</code> to hide the <code>/install-github-app</code> command.
<code>DISABLE_INTERLEAVED_THINKING</code>	Set to <code>1</code> to prevent sending the interleaved-thinking beta header.
<code>DISABLE_LOGIN_COMMAND</code>	Set to <code>1</code> to hide the <code>/login</code> command.
<code>DISABLE_LOGOUT_COMMAND</code>	Set to <code>1</code> to hide the <code>/logout</code> command.
<code>DISABLE_PROMPT_CACHING</code>	Set to <code>1</code> to disable prompt caching for all models (takes precedence over per-model settings).
<code>DISABLE_PROMPT_CACHING_FABLE</code>	Set to <code>1</code> to disable prompt caching for Fable models.
<code>DISABLE_PROMPT_CACHING_HAIKU</code>	Set to <code>1</code> to disable prompt caching for the default Haiku model, wherever it runs.

<code>DISABLE_PROMPT_CACHING_OPUS</code>	Set to <code>1</code> to disable prompt caching for the default Opus model.
<code>DISABLE_PROMPT_CACHING_SONNET</code>	Set to <code>1</code> to disable prompt caching for the default Sonnet model.
<code>DISABLE_TELEMETRY</code>	Set to any non-empty value, such as <code>1</code> , to opt out of telemetry. **Setting it to <code>0</code> or <code>false</code> still...
<code>DISABLE_UPDATES</code>	Set to <code>1</code> to block all updates including manual <code>claude update</code> and <code>claude install</code> .
<code>DISABLE_UPGRADE_COMMAND</code>	Set to <code>1</code> to hide the <code>/upgrade</code> command.
<code>DO_NOT_TRACK</code>	Set to <code>1</code> to opt out of telemetry, with the same effect as <code>DISABLE_TELEMETRY</code> , including on feature-flag...
<code>ENABLE_BETA_TRACING_DETAILED</code>	Set to <code>1</code> , and set <code>BETA_TRACING_ENDPOINT</code> to your OTLP/HTTP collector endpoint, to turn on detailed beta...
<code>ENABLE_CLAUDEAI_MCP_SERVERS</code>	Set to <code>false</code> to stop Claude Code from fetching claude.ai MCP servers.
<code>ENABLE_PROMPT_CACHING_1H</code>	Set to <code>1</code> to request a 1-hour prompt cache TTL instead of the default 5 minutes.
<code>ENABLE_PROMPT_CACHING_1H_BEDROCK</code>	Deprecated.
<code>ENABLE_TOOL_SEARCH</code>	Controls MCP tool search.
<code>FALLBACK_FOR_ALL_PRIMARY_MODELS</code>	Set to any non-empty value, such as <code>1</code> , to make Claude Code stop retrying on repeated overload errors for...
<code>FORCE_AUTOUPDATE_PLUGINS</code>	Set to <code>1</code> to force plugin auto-updates even when the main auto-updater is disabled via <code>DISABLE_AUTOUPDATER</code> .
<code>FORCE_HYPERLINK</code>	Set to <code>1</code> to enable clickable OSC 8 hyperlinks when your terminal supports them but isn't auto-detected,...
<code>FORCE_PROMPT_CACHING_5M</code>	Set to <code>1</code> to force the 5-minute prompt cache TTL even when 1-hour TTL would otherwise apply.
<code>HTTP_PROXY</code>	Specify HTTP proxy server for network connections.
<code>HTTPS_PROXY</code>	Specify HTTPS proxy server for network connections.
<code>IS_DEMO</code>	Set to any non-empty value, such as <code>1</code> , to enable demo mode: hides your email and organization name from...
<code>MAX_MCP_OUTPUT_TOKENS</code>	Maximum number of tokens allowed in MCP tool responses (default: 25000).
<code>MAX_STRUCTURED_OUTPUT_RETRIES</code>	Number of attempts Claude Code allows when the model's response fails validation against the...
<code>MAX_THINKING_TOKENS</code>	Fixed token budget for extended thinking.
<code>MCP_CLIENT_SECRET</code>	OAuth client secret for MCP servers that require pre-configured credentials.
<code>MCP_CONNECTION_NONBLOCKING</code>	Controls whether startup waits for MCP servers to connect before the first query.
<code>MCP_CONNECT_TIMEOUT_MS</code>	How long blocking MCP startup waits, in milliseconds, for the connection batch before snapshotting the...
<code>MCP_DISCOVERY_CACHE</code>	Turns the MCP discovery cache on or off.
<code>MCP_DISCOVERY_CACHE_MAX_STALE_S</code>	Maximum age, in seconds, of a discovery-cache entry (default: 14400, or 4 hours).
<code>MCP_DISCOVERY_CACHE_STRIKES</code>	At a start where a discovery-cache entry is older than <code>MCP_DISCOVERY_CACHE_TTL_S</code> , Claude Code refreshes...
<code>MCP_DISCOVERY_CACHE_TTL_S</code>	Seconds for which Claude Code uses a discovery-cache entry without refreshing it (default: 900).

<code>MCP_OAUTH_CALLBACK_PORT</code>	Fixed port for the OAuth redirect callback, as an alternative to <code>--callback-port</code> when adding an MCP...
<code>MCP_PROTOCOL_NEGOTIATION</code>	On the v2 MCP client runtime only, whether Claude Code probes servers for MCP protocol revision...
<code>MCP_REMOTE_SERVER_CONNECTION_BATCH_SIZE</code>	Maximum number of remote MCP servers (HTTP/SSE) to connect in parallel during startup (default: 20).
<code>MCP_SDK_GENERATION</code>	Pin which MCP client runtime this process connects to MCP servers with: <code>v1</code> , built on MCP TypeScript SDK...
<code>MCP_SERVER_CONNECTION_BATCH_SIZE</code>	Maximum number of local MCP servers (stdio) to connect in parallel during startup (default: 3).
<code>MCP_TIMEOUT</code>	Timeout in milliseconds for MCP server startup (default: 30000, or 30 seconds).
<code>MCP_TOOL_TIMEOUT</code>	Timeout in milliseconds for MCP tool execution (default: 100000000, about 28 hours).
<code>NO_PROXY</code>	List of domains and IPs to which requests will be directly issued, bypassing proxy.
<code>OTEL_ATTRIBUTE_VALUE_LENGTH_LIMIT</code>	Standard OpenTelemetry SDK limit on attribute value length.
<code>OTEL_LOG_ASSISTANT_RESPONSES</code>	Set to <code>1</code> to include the model's response text on <code>assistant_response</code> OpenTelemetry log events.
<code>OTEL_LOG_MANAGED_SETTINGS</code>	Set to <code>1</code> to add the redacted managed settings, and a SHA-256 digest of the settings before redaction, to...
<code>OTEL_LOG_RAW_API_BODIES</code>	Emit Anthropic Messages API request and response JSON as <code>api_request_body</code> / <code>api_response_body</code> log events.
<code>OTEL_LOG_TOOL_CONTENT</code>	Set to <code>1</code> to include tool content in the <code>tool.output</code> OpenTelemetry span event.
<code>OTEL_LOG_TOOL_DETAILS</code>	Set to <code>1</code> to include tool input arguments; MCP server names; user-authored workflow names; raw error...
<code>OTEL_LOG_USER_PROMPTS</code>	Set to <code>1</code> to include user prompt text in OpenTelemetry traces and logs.
<code>OTEL_METRICS_INCLUDE_ACCOUNT_UUID</code>	Set to <code>false</code> to exclude account UUID from metrics attributes (default: included).
<code>OTEL_METRICS_INCLUDE_ENTRYPPOINT</code>	Set to <code>true</code> to include the session entrypoint in metrics attributes (default: excluded).
<code>OTEL_METRICS_INCLUDE_REPOSITORY</code>	Set to <code>true</code> to tag OpenTelemetry metrics and events with <code>vcs.*</code> attributes identifying the session's...
<code>OTEL_METRICS_INCLUDE_RESOURCE_ATTRIBUTES</code>	As of v2.1.161, Claude Code attaches <code>OTEL_RESOURCE_ATTRIBUTES</code> keys to metric datapoint labels.
<code>OTEL_METRICS_INCLUDE_SESSION_ID</code>	Set to <code>false</code> to exclude session ID from metrics attributes (default: included).
<code>OTEL_METRICS_INCLUDE_VERSION</code>	Set to <code>true</code> to include Claude Code version in metrics attributes (default: excluded).
<code>SLASH_COMMAND_TOOL_CHAR_BUDGET</code>	Override the character budget for skill metadata shown to the Skill tool.
<code>TASK_MAX_OUTPUT_LENGTH</code>	Removed in v2.1.277 and now a no-op, together with the <code>TaskOutput</code> tool it sized.
<code>USE_BUILTIN_RIPGREP</code>	Set to <code>0</code> to use system-installed <code>rg</code> instead of <code>rg</code> included with Claude Code.

<code>VERTEX_REGION_CLAUDE_3_5_HAIKU</code>	Override region for Claude 3.5 Haiku when using Google Cloud's Agent Platform.
<code>VERTEX_REGION_CLAUDE_3_5_SONNET</code>	Override region for Claude 3.5 Sonnet when using Google Cloud's Agent Platform.
<code>VERTEX_REGION_CLAUDE_3_7_SONNET</code>	Override region for Claude 3.7 Sonnet when using Google Cloud's Agent Platform.
<code>VERTEX_REGION_CLAUDE_4_0_OPUS</code>	Override region for Claude Opus 4 when using Google Cloud's Agent Platform.
<code>VERTEX_REGION_CLAUDE_4_0_SONNET</code>	Override region for Claude Sonnet 4 when using Google Cloud's Agent Platform.
<code>VERTEX_REGION_CLAUDE_4_1_OPUS</code>	Override region for Claude Opus 4.1 when using Google Cloud's Agent Platform.
<code>VERTEX_REGION_CLAUDE_4_5_OPUS</code>	Override region for Claude Opus 4.5 when using Google Cloud's Agent Platform.
<code>VERTEX_REGION_CLAUDE_4_5_SONNET</code>	Override region for Claude Sonnet 4.5 when using Google Cloud's Agent Platform.
<code>VERTEX_REGION_CLAUDE_4_6_OPUS</code>	Override region for Claude Opus 4.6 when using Google Cloud's Agent Platform.
<code>VERTEX_REGION_CLAUDE_4_6_SONNET</code>	Override region for Claude Sonnet 4.6 when using Google Cloud's Agent Platform.
<code>VERTEX_REGION_CLAUDE_4_7_OPUS</code>	Override region for Claude Opus 4.7 when using Google Cloud's Agent Platform.
<code>VERTEX_REGION_CLAUDE_4_8_OPUS</code>	Override region for Claude Opus 4.8 when using Google Cloud's Agent Platform.
<code>VERTEX_REGION_CLAUDE_5_5_OPUS</code>	Override region for Claude Opus 5.5 when using Google Cloud's Agent Platform.
<code>VERTEX_REGION_CLAUDE_5_5_SONNET</code>	Override region for Claude Sonnet 5.5 when using Google Cloud's Agent Platform.
<code>VERTEX_REGION_CLAUDE_5_OPUS</code>	Override region for Claude Opus 5 when using Google Cloud's Agent Platform.
<code>VERTEX_REGION_CLAUDE_5_SONNET</code>	Override region for Claude Sonnet 5 when using Google Cloud's Agent Platform.
<code>VERTEX_REGION_CLAUDE_FABLE_5</code>	Override region for Claude Fable 5 when using Google Cloud's Agent Platform.
<code>VERTEX_REGION_CLAUDE_FABLE_5_1</code>	Override region for Claude Fable 5.1 when using Google Cloud's Agent Platform.
<code>VERTEX_REGION_CLAUDE_HAIKU_4_5</code>	Override region for Claude Haiku 4.5 when using Google Cloud's Agent Platform.
<code>VERTEX_REGION_CLAUDE_HAIKU_5_5</code>	Override region for Claude Haiku 5.5 when using Google Cloud's Agent Platform.
<code>ANTHROPIC_API_KEY</code> , <code>AWS_SECRET_ACCESS_KEY</code>	Removes it.
<code>NPM_TOKEN</code> , <code>DB_PASSWORD</code>	Removes it, because the name looks like a credential.
<code>DATABASE_URL</code> that contains a password`	Removes it, because the value looks like a credential.
<code>PIP_INDEX_URL</code> or <code>NPM_CONFIG_REGISTRY</code> that contains a password`	Keeps the URL and cuts the username and password from it.

<code>GITHUB_TOKEN</code> , <code>GH_TOKEN</code> , <code>GH_ENTERPRISE_TOKEN</code> , <code>GITHUB_ENTERPRISE_TOKEN</code>	Leaves it in place, so that <code>gh</code> and scripts that call the GitHub API keep working.
<code>HTTP_PROXY</code> , <code>HTTPS_PROXY</code>	Leaves it in place, including a username and password in the URL. The sandbox can set these variables...
<code>GIT_CONFIG_COUNT</code> , <code>GIT_CONFIG_KEY_<n></code> , <code>GIT_CONFIG_VALUE_<n></code>	Leaves it in place, whatever it holds.

Glossary

Agent	An AI that can take actions, such as reading and editing files or running commands, in a loop until a task is finished, rather than only replying with text.
Agentic loop	The cycle an agent repeats: gather context, plan, act, check the result.
Auto mode	A permission mode where a separate safety classifier reviews Claude's actions instead of asking you about each one.
Bypass permissions	A permission mode (<code>--dangerously-skip-permissions</code>) that skips all checks. Meant only for isolated, disposable environments.
Checkpoint	An automatic snapshot taken before Claude edits files, which you can return to with <code>/rewind</code> .
Classifier	A second AI model that judges whether an action in auto mode matches what you asked for.
CLAUDE.md	A Markdown file of instructions Claude reads at the start of every session.
CLI	Command-line interface: a program you control by typing commands in a terminal.
Context window	The amount of conversation and file content Claude can consider at once.
Current directory	The folder your terminal is "in." Claude Code works on the files in this folder. Also called the working directory.
Diff	A view showing exactly which lines of a file were added, removed, or changed.
Git	A tool that records the history of your files so you can review changes and go back in time.
Hook	A command you configure to run automatically at a set moment, such as after every file edit.

Markdown	A simple text format that uses symbols like <code>#</code> and <code>-</code> for headings and lists.
MCP	Model Context Protocol, an open standard for connecting AI tools to other apps and data.
MCP server	A connector that gives an AI new tools, such as access to Notion, GitHub, or a database.
Model	The specific AI “brain” doing the work, e.g. Opus (most capable), Sonnet (balanced), Haiku (fastest).
PATH	The list of folders your terminal searches when you type a program’s name.
Permission mode	The setting that decides which actions Claude can take without asking you.
Plan mode	A permission mode in which Claude researches and proposes a plan but makes no edits until you approve.
Plugin	An installable bundle of skills, subagents, hooks, and MCP servers.
Prompt	The request you type to Claude. Also, the terminal’s “ready for input” marker.
Prompt injection	Hidden instructions in a web page, file, or tool result that try to trick an AI into doing something you didn’t ask for.
Sandbox	An isolated environment that limits which files and network addresses a command can reach.
Skill	A saved set of instructions, in a <code>SKILL.md</code> file, that Claude can use automatically or that you can run as a <code>/command</code> .
Subagent	A specialist helper with its own instructions and tools that Claude can delegate work to.
Terminal	A text window for typing commands to your computer. Also called a shell, console, or command prompt.
Token	A small chunk of text, roughly three-quarters of a word, used to measure AI usage.
Worktree	A separate working copy of a Git repository, so parallel tasks don’t step on each other.