COLOUR KEY: Keyboard · Slash cmd · Workflow · Skill/agent · CLI/flag · Config/env · ★ New in 2026 · Removed/deprecated

Getting the Most Out of Claude Code

A practical guide and reference — verified July 16 2026 against CLI v2.1.211 · Compiled by Douglas Mun

Claude Code is an agentic coding assistant — it reads your repo, runs commands, edits files, and manages git workflows within the permission boundaries you set. The gap between casual and expert use is narrow but compounds: practitioners plan before they build, give Claude durable project memory, isolate risky work in worktrees, and offload repeated thinking to skills, plugins, and subagents.

Entries in section colour are stable. ★ marks features new in 2026 (v2.1.160+). Struck-through entries are removed/deprecated. Start non-trivial work in plan mode (Shift+Tab or /plan). Put durable context in CLAUDE.md. Raise effort with /effort xhigh on Opus 4.8. Install an LSP plugin so Claude sees type errors live.

SESSION & CONTEXT

`/clear` Clear history — aliases: /reset, /new
`/compact [instr]` Compact conv. with optional focus
`/context` Visualise context usage as a grid
`/copy [N]` Copy last (or Nth) response
`/diff` Interactive diff — uncommitted + per-turn
`/export [file]` Export conversation as plain text
`/rewind` Rewind conv/code (alias: /checkpoint)
`/branch [name]` Branch conversation (alias: /fork)
`/resume [session]` Resume by ID/name (alias: /continue)
`/rename [name]` Rename session; shows on prompt bar
`/recap` ★ Show session context recap
`/btw <question>` Side question — no context cost

Model & Effort

`/model [model]` Pick/change model; ←→ adjusts effort
`/effort [level]` low/med/high/xhigh★/max/auto
`/fast [on|off]` Toggle fast mode

Display & Config

`/config` Open settings (alias: /settings)
`/theme` ★ Change theme — custom themes
`/color [color]` Set prompt-bar colour for session
`/tui [fullscreen]` ★ Flicker-free fullscreen render
`/focus` ★ Toggle focus view
`/statusline` ★ Configure shell prompt status line
`/keybindings` Open/create keybindings config
`/terminal-setup` Configure Shift+Enter shortcut
`/permissions` Manage allow/ask/deny rules
`/status` Settings — version, model, network

TOOLS & INTEGRATION

`/init` Initialize project CLAUDE.md
`/memory` Edit CLAUDE.md; manage auto-memory
`/mcp` Manage MCP server connections
`/hooks` View hook configurations
`/skills` List skills — t sorts by tokens
`/agents` Agents view (setup wizard removed)
`/plugin [list]` ★ Plugin mgr; list --enabled/--disabled
`/chrome` Configure Claude in Chrome
`/cd <path>` ★ Change session working directory
`/dataviz` ★ Chart + dashboard design skill
`/add-dir <path>` Add working directory
`/install-github-app` ★ Set up Claude GitHub Actions
`/install-slack-app` ★ Install Claude Slack app (OAuth)
`/web-setup` ★ Connect GitHub to Code on web

Workflows & Agentic

`/plan [desc]` Enter plan mode with optional task
`/batch <instr>` 5–30 worktree agents [Skill]
`/simplify [focus]` 3-agent parallel review [Skill]
`/debug [desc]` Debug logging + analysis [Skill]
`/loop [interval]` Recurring task; self-paces [Skill]
`/workflows` ★ Deterministic multi-agent orchestration
`/schedule [desc]` ★ Create/run routines
`/ultraplan <prompt>` ★ Cloud plan editor → web UI
`/code-review ultra` ★ Cloud multi-agent review
`/autofix-pr [prompt]` ★ Web agent auto-fix PR CI
`/security-review` ★ Scan branch diff for vulns
`/team-onboarding` ★ Replayable guide from history
`/claude-api` Load API reference [Skill]

Remote & Sessions

`/remote-control` Enable remote control (alias: /rc)
`/teleport /tp` ★ Pull web session → terminal
`/remote-env` ★ Configure default remote env
`/voice` Toggle push-to-talk dictation
`/desktop /app` Continue in Desktop app
`/mobile /ios` QR code for Claude mobile app
`/sandbox` ★ Toggle sandbox mode
`/tasks` ★ List bg tasks (alias: /bashes)

UTILITY

`/usage` ★ Plan limits + rate status
`/usage-credits` ★ Request more credits from admin
`/stats` Opens /usage → Stats tab
`/cost` Opens /usage → Cost tab
`/extra-usage` Keep working past rate limits
`/insights` Report on patterns + friction
`/doctor /checkup` Full setup checkup; f = auto-fix
`/powerup` ★ Interactive tutorials + demos
`/release-notes` Changelog — version picker
`/feedback /bug` Submit feedback
`/help` Show help and commands
`/exit /quit` Exit the CLI
`/login /logout` Sign in / out

Removed / Deprecated

~~`/pr-comments`~~ Removed v2.1.91 — ask Claude
~~`/vim`~~ Removed v2.1.92 — /config → Editor
~~`/review`~~ Deprecated — code-review plugin
~~`/tag`~~ Removed in v2.1.92
~~`/ultrareview`~~ Deprecated → /code-review ultra
~~`/agents-wizard`~~ Setup wizard removed v2.1.198

★ QUICK TIPS

Type /s to see every command filtered live
Skills (bundled or user-made) share the same UI as built-in commands
Some commands are platform-dependent
Plugin commands are namespaced: /<plugin-name>:<command>

COLOUR KEY: Keyboard · Slash cmd · Workflow · Skill/agent · CLI/flag · Config/env · ★ New in 2026 · Removed/deprecated

GENERAL CONTROLS

Ctrl+C Cancel input / generation
Ctrl+D Exit session (EOF)
Ctrl+L Clear prompt + force redraw
Ctrl+O Toggle transcript viewer
Ctrl+R Reverse search history
Ctrl+G Open prompt in editor
Ctrl+B Background task (tmux: twice)
Ctrl+T Toggle task list
Ctrl+V Paste image → [Image #N]
Esc Esc Rewind / summarize
↑ ↓ Cursor / history navigation

Text Editing

Ctrl+A Start of current line
Ctrl+E End of current line
Ctrl+K Delete to end of line
Ctrl+U Delete to start of line
Ctrl+W Delete previous word
Ctrl+Y Paste deleted text (yank)
Alt+B / Alt+F Back / forward one word ■

Multiline Input

\+Enter Universal — any terminal
Ctrl+J Universal control seq
Shift+Enter iTerm2/WezTerm/Ghostty/Kitty
Option+Enter macOS Option-as-Meta

MODE & NAVIGATION

Permission & Model Modes

Shift+Tab Cycle permission modes
Alt+M Cycle modes (alt)
Option+P / Alt+P Switch model
Option+T / Alt+T Toggle thinking
Option+O / Alt+O Toggle fast mode

Transcript Viewer (Ctrl+O)

Ctrl+E Toggle show all content
/ Search transcript
n / N Next / previous match
[Dump to scrollback
v Open in $VISUAL/$EDITOR
q Exit transcript

Voice Dictation

Space Hold or tap to dictate
/voice Tap → push-to-talk mode

VIM EDITOR MODE

Enable: /config → Editor mode

Esc NORMAL mode
i / a Insert before / after
h j k l Move left/down/up/right
w e b Word next/end/prev
0 $ ^ Line start / end / non-blank
gg G Top / bottom of input
x dd D Del char / line / to end
yy p Yank / paste
u . Undo / repeat

■ macOS OPTION-AS-META SETUP

Alt+B, Alt+F, Alt+M, Alt+P, Alt+T need Option configured as Meta:
iTerm2: Profiles → Keys → Left Option = Esc+
Terminal.app: check “Use Option as Meta Key”
VS Code: terminal.integrated.macOptionIsMeta:true

Beyond the Cheat Sheet — Tips to Save Tokens and Work Smarter

1. Know where tokens go. Every turn sends the full conversation, active tool schemas, loaded CLAUDE.md, and system prompt. Run /context to see the breakdown; check /usage after expensive ops. Find the single biggest offender rather than micro-optimising.

2. Plan before you spend. Enter plan mode for anything non-trivial — Claude researches and proposes in one pass. Use /compact focus on <topic> proactively. Put durable facts in CLAUDE.md so they survive compaction.

3. Match model & effort. Opus 4.8 at xhigh is extraordinary for complex agentic coding; routine edits run fine on Sonnet 5 (default, 1M context) at medium. Subscription users get a 1-hour cache TTL automatically.

4. Delegate — don't clutter. Spawn the Explore subagent (read-only, inherits session model capped at Opus). Subagents now run in the background by default. Use /btw for side questions. For parallel work, launch --worktree sessions.

COLOUR KEY: Keyboard · Slash cmd · Workflow · Skill/agent · CLI/flag · Config/env · ★ New in 2026 · Removed/deprecated

CLI COMMANDS

`claude` Start interactive session
`claude "query"` Session with initial prompt
`claude -p "query"` Headless / print mode
`claude -c` Continue most recent conv.
`claude -r "name"` Resume by name or ID
`claude update` Update to latest version
`claude install [ver]` ★ Install specific / stable
`claude auth login` ★ Sign in (--console for API)
`claude auth status` ★ Auth status (--text = human)
`claude agents` ★ List configured subagents
`claude setup-token` ★ Long-lived token for CI
`claude remote-control` ★ Start remote-control server
`claude mcp` Configure MCP servers
`claude plugin` ★ Manage plugins

Key CLI Flags

`--model` sonnet / opus / full name
`--permission-mode` default/acceptEdits/plan/auto★/bypass
`--effort` low/med/high/xhigh★/max
`--output-format` text / json / stream-json
`--max-turns N` Limit agentic turns
`--max-budget-usd N` Cost cap per session
`--worktree / -w` ★ Isolated git worktree
`--tmux` ★ Create tmux pane for worktree
`--bare` ★ Faster startup — skip plugins
`--agent <name>` ★ Specify agent for session
`--agents {JSON}` ★ Define agents dynamically
`--from-pr N` ★ Resume PR-linked sessions
`--fork-session` ★ New ID when resuming
`--name / -n "x"` ★ Set session display name
`--remote "task"` Create web session
`--teleport` ★ Pull web → local terminal
`--mcp-config <file>` ★ Load MCP from JSON
`--settings <file>` ★ Load extra settings
`--json-schema {..}` ★ Validated JSON output
`--forward-subagent-text` ★ Subagent text/thinking in stream
`--safe-mode` ★ Start w/ all customizations off
`--dangerously-skip-permissions` Bypass ALL checks
`--verbose` Full turn-by-turn logs
`--version / -v` Show version

CONFIG & ENV VARIABLES

Authentication

`ANTHROPIC_API_KEY` API key (overrides sub)
`CLAUDE_CODE_OAUTH_TOKEN` Long-lived OAuth for CI
`ANTHROPIC_AUTH_TOKEN` Custom Authorization header
`ANTHROPIC_BASE_URL` Proxy / gateway endpoint

Providers

`CLAUDE_CODE_USE_BEDROCK` Use AWS Bedrock
`CLAUDE_CODE_USE_VERTEX` Use Google Vertex AI
`CLAUDE_CODE_USE_FOUNDRY` ★ Use Microsoft Foundry
`CLAUDE_CODE_USE_MANTLE` ★ Use Bedrock Mantle
`AWS_BEARER_TOKEN_BEDROCK` Bedrock API key

Behaviour & Features

`DISABLE_AUTOUPDATER` Disable auto-updates
`DISABLE_UPDATES` ★ Block ALL updates
`CLAUDE_CODE_ENABLE_AUTO_MODE` ★ Opt into auto mode (BR/VX)
`CLAUDE_CODE_EXPERIMENTAL_AGENT_TEAMS` ★ Enable agent teams (=1)
`CLAUDE_CODE_DISABLE_1M_CONTEXT` ★ Block 1M context models
`CLAUDE_CODE_DISABLE_MOUSE_CLICKS` ★ Disable mouse clicks
`CLAUDE_CODE_DISABLE_ALTERNATE_SCREEN` ★ Inline — no fullscreen
`CLAUDE_CODE_RETRY_WATCHDOG` ★ Raise retry on stalls
`CLAUDE_CODE_FORWARD_SUBAGENT_TEXT` ★ Subagent text in stream-json
`CLAUDE_CODE_SAFE_MODE` ★ Start w/ customizations off
`CLAUDE_CODE_PROCESS_WRAPPER` ★ Corporate launcher wrap
`CLAUDE_CLIENT_PRESENCE_FILE` ★ Client presence marker
`MAX_THINKING_TOKENS` Thinking budget (0=off)
`MAX_MCP_OUTPUT_TOKENS` MCP tool response cap
`CLAUDE_CODE_MAX_RETRIES` API retry count (cap 15)
`ENABLE_PROMPT_CACHING_1H` ★ Request 1-hour cache TTL
`CLAUDE_CONFIG_DIR` Override config directory
`CLAUDE_ENV_FILE` Sourced before each Bash cmd

PERMISSION MODES & HOOKS

Permission Modes (Shift+Tab)

`default` Prompt on risky — "Manual" in UI
`acceptEdits` Auto-allow edits; prompt on Bash
`plan` Read-only research / planning
`auto` ★ Classifier allows safe / denies risky
`dontAsk` Deny silently, no prompt
`bypassPermissions` Skip ALL checks (dangerous)

Hook Events (lifecycle)

`SessionStart` Session begins or resumes
`SessionEnd` Session terminates
`PostSession` ★ After session — self-hosted runners
`UserPromptSubmit` Before prompt processed
`UserPromptExpansion` ★ Before user cmd → prompt
`PreToolUse` Before tool call (can block)
`PostToolUse` After tool call succeeds
`PostToolUseFailure` ★ After tool call fails
`PostToolBatch` ★ After parallel batch resolves
`PermissionRequest` Permission dialog appears
`PermissionDenied` ★ Auto-mode denial (can retry)
`SubagentStart` ★ Subagent spawned
`SubagentStop` ★ Finishes — can add context
`TaskCreated` ★ Via TaskCreate
`TaskCompleted` ★ Task marked complete
`Stop` ★ Finishes — can add context
`StopFailure` ★ Turn ended on API error
`PreCompact` ★ Before context compaction
`PostCompact` ★ After compaction completes
`Notification` Claude sends notification
`CwdChanged` ★ Working directory changes
`FileChanged` ★ Watched file changes on disk
`InstructionsLoaded` ★ CLAUDE.md / rules loaded

COLOUR KEY: Keyboard · Slash cmd · Workflow · Skill/agent · CLI/flag · Config/env · ★ New in 2026 · Removed/deprecated

CODE INTELLIGENCE (LSP)

Install binary FIRST, then plugin

`typescript-lsp` TS / JavaScript
`pyright-lsp` Python
`rust-analyzer-lsp` Rust
`gopls-lsp` Go
`clangd-lsp` C / C++
`csharp-lsp` C#
`jdtls-lsp` Java
`kotlin-lsp` Kotlin
`swift-lsp` Swift
`lua-lsp` Lua
`php-lsp` PHP

What Claude Gains

Auto diagnostics Errors after every edit
Go to definition Jump across files
Find references All usages of a symbol
Hover types Type info without compiling

OFFICIAL PLUGINS

Marketplace: claude-plugins-official

`github` PRs, issues, repos, Actions
`gitlab` MRs, issues, pipelines

Project / Design

`atlassian` Jira + Confluence
`asana` Tasks via Asana MCP
`linear` Issues, cycles
`notion` Pages, databases, search

Infra / Comms

`figma` Files, components, designs
`vercel` Deploy, inspect projects
`firebase` Firestore, Auth, Hosting
`supabase` Postgres, Auth, Edge Fns
`slack` Post messages, read channels
`sentry` Errors, issues, replays

Dev Workflows

`commit-commands` Git commit/push/PR
`pr-review-toolkit` PR review agents
`code-review` Replaces deprecated /review
`agent-sdk-dev` Build w/ Agent SDK
`plugin-dev` Create & debug plugins

PLUGIN MANAGEMENT

CLI Commands

`claude plugin install <n>` Install from marketplace
`claude plugin uninstall <n>` Remove (--keep-data)
`claude plugin enable/disable` Toggle plugin
`claude plugin update <n>` Update to latest
`claude plugin list` List installed (--json)
`claude plugin validate` Check schemas

Plugin Path Variables

`${CLAUDE_PLUGIN_ROOT}` Install dir (changes on update)
`${CLAUDE_PLUGIN_DATA}` Persistent — survives updates
`${user_config.KEY}` User-supplied config value

★ A+ TIPS

/reload-plugins after install = no restart
--plugin-dir <path> loads local plugins
claude plugin validate catches schema errors
$(CLAUDE_PLUGIN_DATA) survives updates
Ctrl+O shows LSP diagnostics indicator

COLOUR KEY: Keyboard · Slash cmd · Workflow · Skill/agent · CLI/flag · Config/env · ★ New in 2026 · Removed/deprecated

MODEL GUIDE

Models (Jul 2026)

Opus 4.8 Top model — complex agentic coding
Sonnet 5 Default · 1M-token context
Haiku 4.5 Fast, cheap — light tasks
Fable 5 ★ Mythos-class model

Effort Levels

low Renames, mechanical edits
medium Routine coding, lookups
high Multi-file changes
xhigh ★ ★ Hard agentic problems (Opus 4.8)
max Deepest reasoning

★ WHAT CHANGED IN v2.0

Sonnet 5 default · Opus 4.8 top · Fable 5
"default" mode now labeled Manual (2.1.200)
Subagents run background by default; nest up to 5 levels; auto-commit/push/PR
/workflows — deterministic orchestration
/ultrareview → /code-review ultra
/agents setup wizard removed (2.1.198)

QUICK WINS

`--bare` Skip plugins/hooks (fast start)
`--max-budget-usd N` Cap headless runs
`Grep/Glob` Prefer over Read for discovery
`MAX_THINKING_TOKENS=0` When thinking doesn't help
`/clear` After a break vs re-caching stale
`/skills t` Sort by tokens — prune heavy

★ TOKEN DISCIPLINE

Every turn re-sends full conversation + tool schemas + CLAUDE.md + system prompt
/context finds the biggest offender fast
/btw reuses parent cache, skips history
Prefer @path/to/file over pasting contents
Avoid switching models mid-conversation — next turn re-reads history uncached

CONFIG FILES

`~/.claude/settings.json` User (global)
`.claude/settings.json` Project (via git)
`.claude/settings.local.json` Local (gitignored)
`~/.claude/CLAUDE.md` Global personal memory
`CLAUDE.md (root)` Project memory (survives compact)
`.claude/rules/*.md` Rule fragments by path

New Settings Keys

`fallbackModel` ★ Up to 3 fallback models
`disableBundledSkills` ★ Turn off bundled skills
`wheelScrollAcceleration…` ★ Mouse-wheel accel toggle
`Tool(param:value)` ★ Permission rule by tool input

★ EXPERT HABITS

Plan before you build — /plan for anything touching more than a couple files
Durable facts → CLAUDE.md, not chat
Codify repeated workflows as skills
Isolate risky work in --worktree sessions
Install an LSP plugin for live type errors

BUILD AN OBSIDIAN VAULT WITH CLAUDE CODE · ★ NEW

An Obsidian vault is just a folder of markdown files — Claude Code edits it natively. Point a session at the vault, give it durable conventions in CLAUDE.md, and codify repeat chores (daily notes, MOCs, backlink cleanup) as skills. No Obsidian plugin required.

Vault Setup

cd into the vault claude --add-dir ~/Vaults/Main (or /cd)
CLAUDE.md at root Vault map, folder rules, tag taxonomy
Frontmatter rules Required keys: title, tags, created
Link hygiene Prefer [[wikilinks]]; never bare paths
Folders as context daily/, moc/, refs/, templates/
Skill: new-note ★ Templated note w/ frontmatter + links
Skill: weave-links ★ Add [[backlinks]] across related notes

Prompt Recipes

"Daily note for today" Generates from templates/daily.md
"Build a MOC for X" Map-of-content from tag/folder scan
"Fix orphan notes" Find no-backlink notes, suggest links
"Normalize tags" Merge #ml/#machine-learning duplicates
"Summarize this folder" Rollup note over refs/ subfolder
"Extract tasks" Pull - [ ] items into a task MOC

★ VAULT TIPS

Add vault CLAUDE.md to .gitignore if the vault is its own git repo — keep memory local
Use @templates/daily.md to reuse a template
Run in plan mode first on large restructures
Glob **/*.md so Claude sees the whole vault