AIINBANGLA · COM

CLAUDE CODE
POWER CHEATSHEET

Slash Commands · Keyboard Shortcuts · MCP
Skills · Agents · CLI Flags · Memory · Config

CLI v2.1+ | Opus / Sonnet / Haiku | Worktrees | 1M Context

AUTHOR

Nasir Uddin Shamim

Creator · Educator · AIinBangla.Com

Compiled September 2026 · Educational reference

How to use this book

Claude Code is Anthropic's agentic coding CLI. It reads your repo, edits files, runs shell commands, talks to MCP servers, and can spawn subagents. This ebook packs the commands, shortcuts, files, and power-user patterns into a printable desk reference.

Type / on an empty prompt to see every command available in YOUR install — plugins, skills, and MCP servers add extra entries. A few commands only appear on Max / Team / certain platforms.

WHAT'S INSIDE

FIRST 10 MINUTES IN A NEW REPO

```
1. cd your-project && claude
2. /login   (if needed) then /status
3. /init    → generate starter CLAUDE.md
4. /memory  → tighten project rules
5. /mcp     → connect GitHub / docs / DB servers
6. /permissions  → set auto-accept vs plan mode
7. Shift+Tab until you land on Plan, then describe the first task
8. /compact when context gets heavy; /rewind if a turn goes wrong
```

DISCLAIMER

"Illegal tips" in the source posts means undocumented or power-user flags — not criminal activity. Always verify against `claude --version` and official docs at code.claude.com. Flags and slash commands ship and change quickly. Do not use --dangerously-skip-permissions outside a sandbox.
Reference site for this edition: AlinBangla.Com · Author: Nasir Uddin Shamim

Keyboard shortcuts

GENERAL CONTROLS

`Ctrl + C`	Cancel input / interrupt generation
`Ctrl + D`	Exit session (EOF on empty prompt)
`Ctrl + L`	Redraw / clear the terminal screen
`Ctrl + O`	Toggle verbose thinking / transcript
`Ctrl + R`	Reverse-search prompt history
`Ctrl + G`	Open current prompt in $EDITOR
`Ctrl + X E`	Open in editor (alias)
`Ctrl + B`	Background the current running task
`Ctrl + T`	Toggle background task list
`Ctrl + V`	Paste image from clipboard
`Ctrl + X K`	Kill background agents (press twice)
`Esc`	Interrupt Claude mid-response
`Esc Esc`	Rewind / checkpoint / summarize menu

MODE SWITCHING

`Shift + Tab`	Cycle permission modes (see page 13)
`Alt + P`	Switch model picker
`Alt + T`	Toggle extended thinking on / off
`Alt + O`	Toggle fast mode
`Ctrl + O`	See thinking in verbose mode

INPUT & PREFIXES

`\ + Enter` / `Ctrl + J`	Insert a newline in the prompt
`/`	Slash-command menu (start of line)
`!`	Run a bash command directly
`@`	File / folder mention + autocomplete
`Tab`	Autocomplete files, commands, @paths
`↑` / `↓`	Cycle prompt history

SESSION PICKER (/resume)

`↑` / `↓`	Navigate sessions	`←` / `→`	Expand / collapse groups
`P`	Preview session	`R`	Rename session
`/`	Search sessions	`A`	Show all projects
`B`	Filter current git branch	`Enter`	Resume selected session

Tip: name sessions early with `/rename auth-refactor` so /resume stays usable after a week of work.

Plan mode, thinking & effort

PLAN MODE

`Shift + Tab`	Normal → Auto-Accept → Plan (and bypass if enabled)
`--permission-mode plan`	Start the session already in plan mode
`/plan [desc]`	Enter plan mode and optionally kick off analysis
`acceptEdits`	Auto-accept file edits; still ask for bash / network

THINKING & EFFORT

`Alt + T`	Toggle extended thinking on / off
`"ultrathink"`	Magic word: max effort for the current turn only
`Ctrl + O`	See the thinking trace in verbose mode
`/effort low|medium|high|max|auto`	Set reasoning budget for the session
`--effort max`	CLI flag equivalent
`CLAUDE_CODE_EFFORT_LEVEL`	Default effort via environment
`MAX_THINKING_TOKENS`	0 disables thinking tokens

CONTEXT MANAGEMENT

`/context`	Visual grid of what is eating the window + tips
`/compact [focus]`	Summarize history; keep CLAUDE.md & skills
`/compact keep the schema, drop UI talk`	Focused compact — pass instructions
`Auto-compact`	Fires around ~95% of the context window
`/autocompact auto|500k`	Tune the auto-compact threshold
`1M context`	Opus 4.6+ on Max / Team / Enterprise
`CLAUDE.md survives compact`	Project memory is re-injected after summary
`/clear`	Wipe conversation; project memory stays
`/btw [question]`	Side Q — does not pollute main context

WHEN TO USE WHICH

Plan first on refactors that touch many files. Effort max + Opus for architecture and hard bugs.
Haiku / fast mode for grep-heavy explore. Compact when answers start forgetting early decisions.
/btw is for “what does this flag mean?” without poisoning the implementation thread.

Slash commands · session & context

Type / at the start of a prompt. Arguments follow the name. Skills can be chained: /skill-a /skill-b do XYZ

`/help`	List every command in this install	`/context`	Visualize window usage
`/clear [name]`	Fresh conversation (aliases /reset /new)	`/diff`	Interactive uncommitted + per-turn diffs
`/compact [focus]`	Summarize to free context	`/todos`	List current TODO items
`/resume [name]`	Reopen a previous session	`/tasks`	Background agents in this session
`/continue`	Alias of /resume last session	`/bashes`	Background bash jobs
`/rename [name]`	Name the current session	`/background [p]`	Detach session as background agent
`/branch [name]`	Fork conversation at this point	`/subtask <task>`	Hand work to a forked subagent
`/fork`	Copy into a background session	`/cd <path>`	Move session to another directory
`/rewind`	Roll conversation and/or code back	`/add-dir <path>`	Grant extra directory access
`/export [file]`	Save transcript to file or clipboard	`/color [c]`	Prompt-bar color (syncs to remote)
`/copy`	Copy last reply	`/theme`	Terminal color theme
`/btw [q]`	Side question, no main-thread cost	`/vim`	Toggle vim input mode
`/cost`	Token usage + spend this session	`/keybindings`	Edit custom key bindings
`/usage`	Plan quota and rate-limit status	`/terminal-setup`	Install Shift+Enter bindings
`/stats`	Streaks, history, model prefs	`/statusline`	Configure status-line UI
`/insights`	Session analytics report	`/privacy-settings`	Privacy preferences
`/status`	Account, model, dir, connectivity	`/release-notes`	Full changelog
`/exit`	Quit the CLI (alias /quit)	`/login` `/logout`	Authenticate or switch accounts

Slash commands · build, review, ops

PROJECT & MEMORY

`/init`	Explore repo and write a starter CLAUDE.md
`/memory`	Open / edit CLAUDE.md files in scope
`/config`	Settings UI (theme, model, editor)
`/model [id]`	Switch model; arrows also tweak effort
`/effort [lvl]`	low · medium · high · max · auto
`/fast [on|off]`	Faster output path where available
`/permissions`	Allow / ask / deny tool patterns
`/sandbox`	Sandboxed bash with FS + network isolation

REVIEW & QUALITY

`/plan [desc]`	Read-only planning pass before edits
`/review`	Code review of current diff (alias /code-review)
`/code-review high 1234`	Review a PR number at a given effort
`/code-review --fix`	Apply review findings
`/code-review ultra`	Multi-agent cloud review
`/simplify`	Bundled skill: 3 parallel review agents
`/security-review`	Scan pending changes for vulns
`/pr-comments [PR]`	Fetch GitHub review comments
`/debug [desc]`	Troubleshoot from the debug log
`/doctor`	Install / MCP / permissions health check

EXTENSIBILITY

`/mcp`	Manage MCP servers + OAuth
`/skills`	List skills (type-to-filter)
`/reload-skills`	Rescan skills without restart
`/agents`	Create / list / edit subagents
`/hooks`	Lifecycle hooks for tool events
`/plugin`	Install / enable / disable plugins
`/reload-plugins`	Hot-reload plugins
`/chrome`	Claude-in-Chrome settings
`/voice`	Push-to-talk dictation (20 languages)
`/loop 5m /review-pr`	Local recurring task (up to ~3 days)
`/schedule`	Cloud scheduled tasks (subscribers)
`/remote-control`	Continue this session from another device
`/teleport`	Pull a claude.ai web session into the CLI
`/feedback  /bug`	Send a report with optional transcript

Bundled skills, workflows & magic words

BUNDLED SKILLS (ship with the CLI)

`/simplify`	Quality pass with 3 parallel review agents
`/batch <instruction>`	Decompose a large change into 5–30 worktrees
`/debug [desc]`	Root-cause from logs / failing tests
`/loop [interval] [task]`	Recurring local scheduler
`/claude-api`	Load Anthropic API + SDK reference into context
`/claude-api migrate`	Update existing API calls to a newer model
`/claude-api cost-optimize`	Profile spend and propose savings

PARALLEL & CLOUD

`/batch migrate src/ to TS`	Plan → approve → one agent per unit + PR
`/background`	Free the terminal; session keeps working
`/fork`	Clone conversation into a background agent
`/autofix-pr [prompt]`	Cloud agent watches CI + review comments
`claude --worktree name`	Isolated git worktree for this session
`isolation: worktree`	Agent frontmatter: run in its own worktree
`/desktop`	Hand off to the Claude Desktop app

MAGIC WORDS IN A NORMAL PROMPT

`ultrathink`	Max reasoning budget for this turn
`think hard / think deeply`	Raise effort without changing /effort
`plan first, don't edit yet`	Soft plan-mode without Shift+Tab
`use the Explore agent`	Force a cheap read-only Haiku scout
`don't write code yet`	Keep Claude in analysis

VOICE MODE

`/voice`	Enable push-to-talk
`Hold Space`	Record; release to send
Languages	EN, ES, FR, DE, ZH, JA, KO, BN-adjacent via multilingual STT, + more
Best for	Walking through a bug out loud, or dumping a messy product brief

`MCP slash pattern` `/mcp__<server>__<prompt> [args]` e.g. `/mcp__github__list_prs`

CLI launch patterns & flags

CORE INVOCATIONS

`claude`	Interactive REPL in the current directory
`claude "fix the login bug"`	Start REPL with an initial prompt
`claude -p "query"`	Print / headless mode — run and exit
`cat log | claude -p "explain"`	Pipe stdin into a one-shot query
`claude -c`	Continue the most recent conversation
`claude -r "auth-refactor"`	Resume a named / id session
`claude -n feature-x`	Set display name at startup
`claude update`	Update the CLI
`claude mcp …`	Configure MCP servers from the shell
`claude --version / --help`	Version and full flag list

KEY FLAGS

`--model opus`	Override model for this run	`--output-format json`	Machine-readable print mode
`--effort max`	Reasoning budget	`--output-format stream-json`	Realtime JSON events
`--permission-mode plan`	Start in a given mode	`--json-schema '{…}'`	Validated structured output
`--add-dir ../lib`	Extra working directory	`--max-turns 8`	Cap agentic loops
`--agent name`	Force a named agent	`--max-budget-usd 5`	Hard spend cap
`--allowedTools …`	Pre-approve tool patterns	`--verbose`	Show tool calls live
`--disallowed-tools …`	Blocklist	`--bare`	No hooks / LSP / plugins
`-w / --worktree`	Isolated git worktree	`--chrome`	Enable Chrome integration
`--mcp-config file.json`	Extra MCP servers	`--remote`	Web / cloud session
`--plugin-dir ./dir`	Load plugins for this run	`--dangerously-skip-permissions`	YOLO — sandbox only

HEADLESS RECIPES

```
claude -p --output-format json "list exported functions in src/" > out.json
claude -p --max-turns 6 --permission-mode acceptEdits "fix failing tests"
claude -c -p "now write the changelog from the last 3 commits"
tail -f app.log | claude -p "page me if you see 5xx bursts"
```

MCP servers

Model Context Protocol connects Claude to GitHub, databases, browsers, Slack, Sentry, and custom tools.

TRANSPORTS

`--transport http`	Remote HTTP — recommended for hosted servers
`--transport stdio`	Local process spawned by Claude
`--transport sse`	Remote Server-Sent Events

SCOPES

`Local ~/.claude.json (or .claude.jso`	You only — not committed
`Project .mcp.json`	Shared with the team via git
`User ~/.claude.json`	Global across all projects

SHELL COMMANDS

`claude mcp add <name> -- <cmd> [args]`	Add stdio server (global default)
`claude mcp add --scope project <name>`	Project-scoped
`claude mcp add --transport http <name>`	Remote HTTP server
`claude mcp add --transport sse <name>`	Remote SSE server
`claude mcp add-json <name> '{…}'`	Add from a JSON blob
`claude mcp list`	All servers + connection status
`claude mcp get <name>`	Show one server's config
`claude mcp logs <name>`	Tail a server's logs
`claude mcp remove <name>`	Remove a server
`claude mcp serve`	Expose Claude Code itself as an MCP server
`/mcp`	Interactive manager + OAuth from inside a session

EXAMPLES

```
claude mcp add filesystem -- npx -y @modelcontextprotocol/server-filesystem ~/projects
claude mcp add github --scope project -- npx -y @modelcontextprotocol/server-github
claude mcp add --transport http linear https://mcp.linear.app/mcp
Inside a chat:  @github:issue/42     or     /mcp__github__create_pr
```

NOTES

maxResultSizeChars can be raised (up to ~500K via _meta) when a server returns huge payloads.
Elicitation: some servers can pause mid-task and ask you a question. Approve OAuth from /mcp.

Memory, files & rules

CLAUDE.MD LOCATIONS

`./CLAUDE.md`	Project — commit this; the whole team inherits it
`~/.claude/CLAUDE.md`	Personal — applies to every project on this machine
`/etc/claude-code/`	Managed / org-wide (enterprise images)
`.claude/rules/*.md`	Project rule files (split by topic)
`~/.claude/rules/*.md`	Personal rule files
`paths: (YAML frontmatter)`	Rule only applies to matching globs
`@path/to/file`	Import another file into CLAUDE.md

AUTO MEMORY

```
~/.claude/projects/<project>/memory/
   MEMORY.md          index
   topic_xxx.md       auto-written topic files (cap ~25KB / ~200 lines)
```

Loaded into context automatically. Survives /compact. Edit with /memory.

WHAT TO PUT IN CLAUDE.MD

Stack + package manager + how to run tests (the command, not a novel).
Repo conventions: folder map, naming, “never edit generated/”.
Review bar: what “done” means (types, tests, no snapshot spam).
People rules: “ask before migrating dependencies”, “don’t push to main”.
Keep it short. A 30-line CLAUDE.md beats a 300-line one nobody reads.
Put secrets in env / MCP auth — never in CLAUDE.md.

MINIMAL CLAUDE.MD SHAPE

```
# Project
Node 20 + pnpm.  Tests: pnpm test.  Lint: pnpm lint.
Do not edit /dist or anything under .next/.
API lives in apps/api; UI in apps/web. Shared types in packages/types.
Before a PR: types + unit tests for any new branch. No drive-by refactors.
```

Skills & agents

SKILL LOCATIONS

`.claude/skills/<name>/SKILL.md`	Project skills — commit and share
`~/.claude/skills/<name>/SKILL.md`	Personal skills — all projects
`plugin skills`	Arrive with installed plugins

SKILL.MD FRONTMATTER

`description`	When Claude should auto-invoke this skill
`allowed-tools`	Tools that skip the permission prompt
`model`	Override model just for this skill
`effort`	Override effort level
`paths: [globs]`	Only relevant for matching files
`context: fork`	Run inside a subagent
`$ARGUMENTS`	Placeholder for user text after /skill
`${CLAUDE_SKILL_DIR}`	Absolute path of the skill folder
``!`cmd` ``	Run a command and inject stdout as context
`plugin bin/`	Ship extra executables for Bash

BUILT-IN AGENTS

`Explore`	Fast read-only scout — typically Haiku
`Plan`	Research pass used by plan mode
`General`	Full toolset for complex implementation
`Bash`	Isolated terminal context

AGENT FRONTMATTER

`permissionMode`	default | acceptEdits | plan | dontAsk | bypass
`isolation: worktree`	Run in its own git worktree
`memory: user | project`	Persist findings
`background: true`	Don't block the parent session
`maxTurns`	Hard cap on agentic loops
`model / effort`	Same overrides as skills

Rule of thumb: if you do it more than once a day, make it a skill. If it needs a different personality, make it an agent.

Config files, env vars & permissions

CONFIG FILES (merged, later wins)

`~/.claude/settings.json`	User settings
`.claude/settings.json`	Project settings — shareable
`.claude/settings.local.json`	Local overrides — gitignore this
`~/.claude.json`	OAuth tokens, MCP, session state
`.mcp.json`	Project MCP server list
`managed-settings.d/`	Drop-in org policy fragments

USEFUL SETTINGS KEYS

`modelOverrides`	Map picker aliases to custom model IDs
`autoMemoryDirectory`	Custom auto-memory folder
`worktree.sparsePaths`	Sparse-checkout only the dirs you need
`sandbox.failIfUnavailable`	Exit if the sandbox cannot start
`hooks`	Conditional PreToolUse / PostToolUse / Stop ...
`PermissionDenied`	Hook fired when a tool is denied
`showThinkingSummaries`	Opt-in thinking summaries
`forceRemoteSettingsRefresh`	Fail closed if org settings stale

ENVIRONMENT VARIABLES

`ANTHROPIC_API_KEY`	API key (or AUTH_TOKEN)	`CLAUDE_CODE_SUBPROCESS_*`	Creds into child processes
`ANTHROPIC_MODEL`	Default model id	`CLAUDE_STREAM_IDLE_TIMEOUT`	Streaming watchdog (default 90s)
`ANTHROPIC_BASE_URL`	Custom endpoint	`MCP_CONNECTION_NONBLOCK`	Don’t block startup on MCP
`CLAUDE_CODE_EFFORT_LEVEL`	low / medium / high / max / auto	`CLAUDECODE=1`	Detect “inside Claude’s shell”
`MAX_THINKING_TOKENS`	0 = thinking off	`CLAUDE_CODE_DISABLE_CRON`	Disable scheduled /loop jobs
`CLAUDE_CODE_MAX_OUTPUT_TOKENS`	Default ~32K	`CLAUDE_CODE_PLUGIN_SEED_DIR`	Colon-separated plugin seeds

Permission modes

Cycle with Shift+Tab. Set at launch with --permission-mode <name>. Persist via settings or /permissions.

`default`	Ask before each tool that isn't pre-allowed. Safest everyday mode.
`acceptEdits`	Auto-accept file edits. Still prompts for bash, network, MCP writes.
`plan`	Read-only. Claude researches and writes a plan; no edits until you leave plan.
`dontAsk`	Deny everything except tools you explicitly allowed. Tight lockdown.
`bypassPermissions`	Skip all prompts. Equals --dangerously-skip-permissions. Sandbox only.

PERMISSION PATTERNS

`Bash(git:*)`	Allow any git subcommand
`Bash(git push:*)`	Allow push (be deliberate)
`Bash(rm -rf:*)`	Almost always deny this
`Read / Edit / Write`	Filesystem tools; scope with globs when you can
`WebFetch / WebSearch`	Network — useful, but noisy on huge crawls

PRACTICAL DEFAULT

Daily coding: acceptEdits. Big unknown refactors: plan until the plan looks right, then acceptEdits.
CI / headless: --allowedTools with a tight list, --max-turns, --max-budget-usd. Never bypass on a laptop that has production credentials in the environment.

Power workflows

GIT WORKTREES

`claude --worktree feature-auth`	Isolated branch + directory per feature
`isolation: worktree`	Agent frontmatter — each agent gets its own tree
`worktree.sparsePaths`	Check out only the folders that agent needs
`/batch …`	Auto-creates many worktrees, one PR each

SESSION POWER MOVES

`claude -c`	Pick up the last thread instantly
`claude -r "name"`	Jump to a named thread
`/btw why is this flaky?`	Side channel — zero main-context cost
`Esc Esc`	Rewind a bad turn instead of fighting it
`/branch try-zustand`	Explore an alternate design without losing the first

HEADLESS / SDK

`claude -p "query"`	One-shot, scriptable
`--output-format json`	Pipe into jq / CI
`--max-budget-usd 3`	Hard stop so a loop cannot melt a card
`cat spec.md | claude -p "implement"`	Spec-driven generation

REMOTE & MULTI-DEVICE

`/remote-control`	(alias /rc) Drive this local session from phone / web
`/teleport`	Pull a claude.ai session down into the terminal
`--remote`	Start a cloud session from the CLI
`/desktop`	Continue in the Claude Desktop app

Field notes from power users

Verify, don't vibe.
After any non-trivial edit: run the test you care about. /simplify then /diff before you commit.

One thread, one job.
/clear between unrelated tasks. Mixing a CSS tweak into an auth refactor burns context and judgment.

Name sessions like PRs.
/rename stripe-webhook-retry beats "session 17" when you /resume on Monday.

Plan is cheaper than undo.
Shift+Tab to Plan for anything that will touch more than three files. Approve the plan, then execute.

Compact with intent.
/compact keep the database schema and the auth decisions, drop the UI bikeshed.

Skills beat prompts.
A 20-line SKILL.md with allowed-tools and a checklist will outperform a 200-line pasted prompt.

MCP is leverage, not decoration.
If you copy-paste from GitHub / Sentry / Figma more than twice a day, add the server.

Worktrees for parallel agents.
/batch or isolation: worktree so two agents cannot fight over the same working tree.

Hooks for guardrails.
PreToolUse can block `rm`, force formatters on Write, or require a ticket id in commit messages.

Budget the context window.
/context when answers get sloppy. Huge CLAUDE.md + 12 MCP servers is how 200K vanishes.

Voice for messy briefs.
/voice + hold Space is faster than typing a two-minute product rant — then switch back to keyboard.

Never skip permissions on a hot laptop.
bypassPermissions belongs in a container with no prod tokens. Period.

Resources

OFFICIAL

```
code.claude.com/docs          Claude Code documentation hub
code.claude.com/docs/en/commands     Full slash-command reference
docs.anthropic.com            CLI flags, SDK, memory, hooks
support.claude.com            Official cheatsheet + power-user tips
modelcontextprotocol.io       MCP protocol + server directory
```

THIS EDITION

Compiled September 2026 by Nasir Uddin Shamim for AlinBangla.Com. Built from community “Illegal Claude Tips Part 41” sheets plus official docs and widely used public cheatsheets current as of Claude Code 2.1.x.

Community sheets move faster than docs. Always type /help and check claude --version before you depend on a flag in production automation.

AlinBangla.Com

Nasir Uddin Shamim

Faceless YouTube systems · AI education in Bangla · Creator tools

`AIinBangla.Com`

Progress feels slow until it suddenly isn’t.