

What is Claude Code?

An agentic CLI tool that reads your codebase, edits files, runs shell commands, and manages git workflows. It works across your terminal, VS Code, JetBrains, a desktop app, and the web.

It is not a chat interface that suggests code.

Claude plans work, writes across multiple files, runs tests, and creates PRs. It can take actions you can't undo — permissions matter.

Installing Claude

```
# Recommended (auto-updates)
curl -fsSL https://c.lau de.a -
i/ ins tall.sh | bash
# macOS Homebrew
brew install --cask claude -code
```

First Run

```
cd your-project
claude # interactive session
claude auth login # if not
authen ticated
```

On first run Claude reads your repo and prompts for auth. Run `/init` inside a session to generate a `CLAUDE.md` project file.

Claude Code CLI Flags

Session Control

```
-c          Continue most recent
           session
-r <name>   Resume session by
           name
--resume <id> Resume session by ID
--bg        Run session in
           background
```

Output and Automation

```
-p "prompt" Non-interactive, print
           and exit
--output-format Machine-readable output
           json
--output-format Stream JSON as it
           stream-json arrives
--max-turns N   Limit agentic turns
```

Tools and Permissions

```
--allowed-tools Comma-separated allow
           list, e.g. Read,Grep
--permission- default | acceptEdits |
mode           auto | bypassPermis-
           sions
--model <na- Override model for this
me>           session
--dangerously- DANGER No prompts —
skip-permi- CI only
ssions
```

Keyboard Shortcuts

```
Esc Esc    Interrupt current action and
           rewind the last message
Ctrl+C     Cancel current generation
           (keeps history)
Shift+Tab  Cycle permission mode: ask
           → auto → bypass
Ctrl+B     Push session to background;
           reclaim terminal
@filename  Include a file's contents inline
           in your message
!'cmd'     Run a bash command and
           inject output into message
↑ / ↓      Navigate message history
/keybi-    Edit shortcuts — stored in
ndings     ~/.claude/keybindings.json
```

Slash Commands

Context management

```
/compact  Compress history to free
           context. Pass a focus hint:
           /compact keep auth module
/clear     Wipe conversation history
           entirely. Start fresh.
/status    Show context usage, session
           ID, model, cost
```

Session navigation



By **Dave Child** (DaveChild)
cheatography.com/davechild/
aloneonahill.com

Not published yet.
 Last updated 23rd June, 2026.
 Page 1 of 3.

Sponsored by **Readable.com**
 Measure your website readability!
<https://readable.com>

Slash Commands (cont)

`/rename` Name the current session (no arg = auto-name)

`/fork` Branch conversation at this point (try alternatives without losing current state)

`/resume` Resume a previous session by name or ID

`/export` Export conversation as plain text

Tools and Config

`/model` Switch model mid-session.

`<name>` Takes effect next message.

`/doctor` Diagnose auth, config, and connection issues

`/init` Generate CLAUDE.md for current project

`/schedule` Create a recurring automated task

`/help` List all current slash commands

Model Aliases

`opus / sonnet / haiku` Short aliases: `/model opus`

`auto` Claude picks model per task

Planning and Flow Commands

`/plan` Ask Claude to produce a plan before writing any code. Review and approve, then Claude executes. Use for any non-trivial change.

`/goal` Set a persistent session goal. Claude references it to stay on track across compacts and model switches. Useful for long sessions.

`/effort` Set reasoning depth: low | normal | high. Default on Opus 4.8 is high. Lower for speed/cost, higher for architecture decisions.

`/check point` Save current conversation state as a named restore point. Useful before risky tool sequences.

`/review` Request a code review of current working state. Add `--fix` to apply findings immediately.

Planning and Flow Commands (cont)

`/workflow` Trigger a dynamic workflow — sequences of steps Claude will walk through. See `~/claude/commands/` for custom workflows.

Typical Flow Example:

```
/goal Refactor the auth module to
o PSR-12
/plan
# review the plan, press y to approve
# Claude writes code
/compact focus on auth module
/review --fix
```

Custom Slash Commands

File Locations

`.claude/commands/` Project command

`~/claude/commands/` Personal command

Each `.md` file becomes a slash command name. Invoked with `/filename`.

Example: `.claude/commands/pr-review.md`

```
Review the current branch diff again.
List bugs and security issues as they arise.
Then summarise in one sentence.
Arguments: $ARGUMENTS
```

Use `!`git diff main`` in the file body to pipe git diff into Claude. Use `@path/ to/file` to inline a file. Use `$(A)` for runtime args: `/pr-review focus on SQL`.



By **Dave Child** (DaveChild)
cheatography.com/davechild/
aloneonahill.com

Not published yet.
Last updated 23rd June, 2026.
Page 2 of 3.

Sponsored by **Readable.com**
Measure your website readability!
<https://readable.com>

CLAUDE.md: Project Instructions

Read at the start of every session. The primary way to give Claude persistent context about your project without repeating yourself.

Locations (in priority order)

./CLAUDE.md	Project root — committed to repo, shared with team
./claude/CLAUDE.md	Project-local, can be gitignored
~/claude/CLAUDE.md	Personal global — applies to all projects

What To Put In It

Stack & conventions	PHP version, framework, coding standard (PSR-12)
Build & test commands	How to run tests, linter, deploy
Off-limits files	Files Claude should never touch
Project structure	Where things live and why
Behaviour rules	Always write tests, never edit migrations

Example CLAUDE.md snippet

```
Stack: PHP 8.3, Laravel 11,
MySQL 8
Standard: PSR-12. Run: ./vendor/bin/pint
Tests: ./vendor/bin/pest —
must pass before commit
Never edit files in /database/migrations/
```

Headless and Pipe Mode

```
# Pipe content as context
cat src/Auth.php | claude -p
"Find security issues"
git diff main | claude -p "Review this diff"
git log --oneline -10 | claude -p "Write release notes"
# JSON output (includes cost, session ID, turn count)
claude -p "list TODOs" --output-format json | jq .
# Read-only: no file writes or bash
claude -p "explain architecture" \
  --allowed-tools Read,Grep
# Multi-step: reuse session across calls
SID=$( claude -p "analyse test failures" \
  --output-format json | jq -r '.session_id')
claude -p "fix the failures" --resume "$SID"
# Append system prompt for specialised context
claude -p "review errors" \
  --append-system-prompt "You are an SRE expert"
```

Use -p for scripts and CI. Claude runs the prompt and exits — no interactive session.

Permissions and Safety

Permission Modes

default

acceptEdits

auto

bypassPermissions

Tool allow/deny in settings

```
# .claude/settings.json
{
  "permissions": {
    "allow": ["Bash(git)", "
te"],
    "deny": ["Bash(rm -rf)"]
  }
}
```

Restrict in CLI

```
claude --allowed-tools Read,Grep
th"
```



By **Dave Child** (DaveChild)
cheatography.com/davechild/
aloeonahill.com

Not published yet.
 Last updated 23rd June, 2026.
 Page 3 of 3.

Sponsored by **Readable.com**
 Measure your website readability!
<https://readable.com>