THE INDEPENDENT FIELD GUIDE

Claude Code, a little closer to hand.

32 selected essentials. Eight practical sections. Original explanations linked to official documentation.
This full snapshot is independent of the interactive page's current filters.

SNAPSHOT 01 · Reviewed 2026-11-13 · Not a CLI version or a live update feed
Availability varies by version, platform and account. Commands are examples, not actions executed by this guide.

Keyboard shortcuts

`Ctrl+C` — Interrupt or clear input

While Claude is working, interrupt the operation. When idle, the first press clears the prompt; another press can exit. Terminal behavior can vary.

```
Ctrl+C
```

Official source: https://code.claude.com/docs/en/interactive-mode

`Ctrl+R` — Search your command history

Start a reverse search through earlier prompt history instead of retyping a useful instruction.

```
Ctrl+R
```

Official source: https://code.claude.com/docs/en/interactive-mode

`Ctrl+O` — Open the transcript viewer

Toggle the detailed transcript view to inspect tool activity and execution context. This is a display control, not a way to undo a tool action.

```
Ctrl+O
```

Official source: https://code.claude.com/docs/en/interactive-mode

`Ctrl+G` — Edit the prompt in your editor

Open the current prompt in your configured external editor. Useful when the instruction needs more room than a terminal input line.

```
Ctrl+G
```

Official source: https://code.claude.com/docs/en/interactive-mode

Slash commands

`/help` — Find your way around

Open the in-session help. Commands available on your own installation depend on its version, platform and account.

```
/help
```

Official source: https://code.claude.com/docs/en/skills

`/clear` — Begin a fresh conversation

Clear the current conversation for a new task while retaining project memory. It does not roll back file changes already made.

```
/clear
```

Official source: https://code.claude.com/docs/en/commands

`/compact` — Summarize a long conversation

Condense conversation context when a session grows long. Review important constraints rather than assuming every detail survives a summary.

```
/compact
```

Official source: https://code.claude.com/docs/en/commands

`/config` — Inspect interactive preferences

Open configuration controls. Different options can be saved at different scopes; check the corresponding settings documentation before sharing configuration.

```
/config
```

Official source: https://code.claude.com/docs/en/settings

Workflows & tips

`Map the codebase` — Start broad, then follow one thread

Original prompt recipe: ask for a repository map before narrowing to one behavior. This text is a prompt, not a built-in slash command.

```
Read this repository and outline its main modules without changing files.
```

Official source: https://code.claude.com/docs/en/common-workflows

`Trace one request` — Ask for evidence, not guesses

Original prompt recipe: follow one concrete request end to end and ask for the files supporting the explanation.

```
Trace one request from input to response. Cite the files you inspected.
```

Official source: https://code.claude.com/docs/en/common-workflows

`Plan a small refactor` — Agree on scope before editing

Original prompt recipe: separate the proposed change from its verification. Review the plan before allowing implementation.

```
Propose a small refactor and explain how you would verify it. Do not edit yet.
```

Official source: https://code.claude.com/docs/en/common-workflows

---

`Understand a failing test` — Begin with the observed failure

Original prompt recipe: include the failing command and relevant output, then ask for a diagnosis before a patch. Review logs for private data first.

```
Explain this failing test and the behavior it expects before suggesting a fix.
```

Official source: https://code.claude.com/docs/en/common-workflows

---

Skills & agents

`SKILL.md` — Package a reusable procedure

A skill is a named set of instructions with supporting files when needed. Its instructions load when used, instead of making every conversation carry the whole procedure.

```
.claude/skills/my-skill/SKILL.md
```

Official source: https://code.claude.com/docs/en/skills

`/my-skill` — Invoke your own installed skill

Placeholder name: this works only after you create or install a skill named my-skill. It is not a bundled command.

```
/my-skill
```

Official source: https://code.claude.com/docs/en/skills

`.claude/agents/` — Keep project subagent definitions

Project subagent definitions belong here; user-level definitions use ~/.claude/agents/. A subagent has its own context and consumes usage, so choose a bounded task.

```
.claude/agents/
```

Official source: https://code.claude.com/docs/en/commands

`/agents` — Locate subagent management guidance

In the reviewed documentation this prints guidance to ask Claude to manage subagents or edit their definition files. Older versions used an interactive management screen.

```
/agents
```

Official source: https://code.claude.com/docs/en/commands

MCP servers

`claude mcp` — Manage tool connections

The MCP command family configures external tool servers. Connections can expose outside data and actions; review the server and its permissions before using it.

```
claude mcp
```

Official source: https://code.claude.com/docs/en/cli-reference

`claude mcp add --transport http` — Add a remote HTTP server

Supply a chosen name and a trusted server URL. Angle-bracket placeholders need replacement. This page neither runs the command nor connects a server.

```
claude mcp add --transport http <name> <url>
```

Official source: https://code.claude.com/docs/en/mcp

`.mcp.json` — Recognize an MCP configuration file

MCP configuration can be supplied as JSON. A remote URL entry needs an explicit transport type; a URL without one is not treated as an HTTP server.

```
.mcp.json
```

Official source: https://code.claude.com/docs/en/mcp

`"type": "http"` — Declare the remote transport

Illustrative connection fragment only. Replace example.com with a server you trust, and place the entry inside the documented configuration structure. Do not paste credentials into public files.

```
{"type":"http","url":"https://example.com/mcp"}
```

Official source: https://code.claude.com/docs/en/mcp

Memory & files

`CLAUDE.md` — Share project instructions

Use project instructions for durable conventions and context that apply to the team. Keep reusable multi-step procedures in skills instead of expanding this file indefinitely.

```
CLAUDE.md
```

Official source: https://code.claude.com/docs/en/memory

`CLAUDE.local.md` — Keep personal project instructions

This is a local project instruction file for one person's preferences. Add it to the project's ignore rules when it should remain private.

```
CLAUDE.local.md
```

Official source: https://code.claude.com/docs/en/memory

`~/.claude/CLAUDE.md` — Set personal cross-project guidance

The home-directory instruction file applies to your own work across projects. The tilde is home-directory notation; resolve it appropriately in your shell.

```
~/.claude/CLAUDE.md
```

Official source: https://code.claude.com/docs/en/memory

`.claude/rules/` — Organize path-scoped guidance

Split larger guidance into focused project rule files. Consult the official syntax when restricting a rule to particular paths.

```
.claude/rules/
```

Official source: https://code.claude.com/docs/en/memory

Config & ENV

`~/.claude/settings.json` — Personal settings across projects

User-level JSON settings affect your work across projects on this machine. They are separate from prose instruction files such as CLAUDE.md.

```
~/.claude/settings.json
```

Official source: https://code.claude.com/docs/en/settings

`.claude/settings.json` — Share settings with a project

Project JSON settings can be committed for teammates. Review values and permissions before sharing them, and keep authentication material outside source control.

```
.claude/settings.json
```

Official source: https://code.claude.com/docs/en/settings

`.claude/settings.local.json` — Try personal project overrides

Local project settings are for one user's overrides. When creating the file by hand, make sure the project's ignore rules exclude it as appropriate.

```
.claude/settings.local.json
```

Official source: https://code.claude.com/docs/en/settings

`CLAUDE_CONFIG_DIR` — Choose a different configuration home

Environment variable for relocating the default Claude configuration directory. Set it using the syntax of your actual shell; this page does not alter your environment.

```
CLAUDE_CONFIG_DIR
```

Official source: https://code.claude.com/docs/en/settings

CLI & flags

`claude -p` — Run a non-interactive query

Pass a prompt in print mode and exit after the response. It still invokes a model and may use enabled tools; it is not an offline text transformation.

```
claude -p "Explain this function"
```

Official source: https://code.claude.com/docs/en/cli-reference

---

`claude --continue` — Continue a recent conversation

Continue the most recent eligible conversation for the current directory. Session eligibility varies by mode and installed version.

```
claude --continue
```

Official source: https://code.claude.com/docs/en/cli-reference

---

`claude --agent` — Choose an existing session agent

Start with a named custom agent that is available in your configuration. Replace the sample name with an installed agent definition.

```
claude --agent my-custom-agent
```

Official source: https://code.claude.com/docs/en/cli-reference

---

`claude --add-dir` — Grant an additional directory

Adds file access to an existing directory. It does not discover most project configuration from that extra directory. Replace the path and review the access you are granting.

```
claude --add-dir <existing-directory>
```

Official source: https://code.claude.com/docs/en/cli-reference

---

Independent reference concept. Not affiliated with Anthropic. Claude Code is an Anthropic product.
Workflow prompt recipes and explanations are original. No newsletter, automatic updater or remote service is connected.