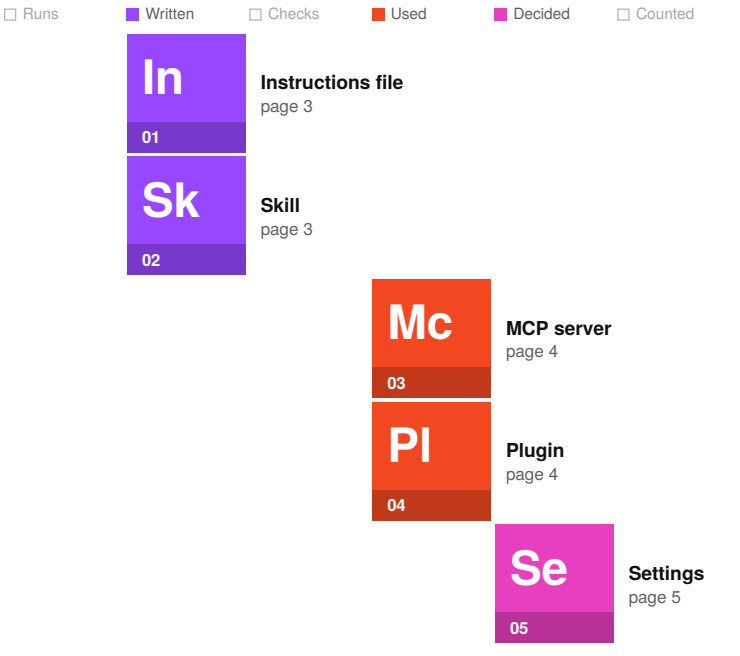
Claude Code

A pocket-sized guide to where Claude Code keeps its setup, as of October 2026.



My friend has a lot of questions.

What can you set up in Claude Code?

Claude Code reads its setup from plain files, some in your home folder and some in the project. There are five kinds: instructions files, skills, MCP servers, plugins and settings.

Each kind can be yours alone, shared with the project, or set by your organization. An Accidental Guide to: Agents, Harnesses, Skills, and Tools explains the basics.

Instructions file	Text added to every session, such as build commands.
Skill	A written procedure, loaded when a task needs it.
MCP server	A program that gives Claude Code tools for another system.
Plugin	A package of skills, MCP servers and hooks, installed together.
Settings	JSON files for permissions, hooks and other behavior.

Instructions file

In
01

Identify by `CLAUDE.md`, or `AGENTS.md`.

What **An instructions file holds what Claude Code should know in every session.** Claude Code reads `AGENTS.md` only when there's no `CLAUDE.md`. An Accidental Guide to: Agents, Harnesses, Skills, and Tools explains instructions files.

So what **Write down what you'd otherwise explain again each session.** Run `/init` to start a `CLAUDE.md`.

Field marks

Loads	Cannot
At every session start.	**Enforce a rule.**

Confused with Se **Settings**
Instructions are guidance the model reads. Settings are rules Claude Code enforces.

Skill

Sk
02

Identify by A folder with a `SKILL.md` file.

What **A skill is a written procedure that Claude Code loads when a task needs it, or when you type its name, such as** `/deploy`**.** An Accidental Guide to: Agent Skills explains skills.

So what Steps needed only for some tasks belong in a skill, so they don't fill every session.

Field marks

Loads	Cannot
Description at startup.	**Add a tool.**

Confused with In **Instructions file**
An instructions file loads in full every session. A skill loads in full only when it's used.

MCP server

Mc
03

Identify by An entry under mcpServers, with a start command or a URL.

What **An MCP server is a separate program that gives Claude Code tools for another system.** Add one with claude mcp add. An Accidental Guide to: Model Context Protocol explains MCP servers.

So what Servers in a project's `.mcp.json` wait for your approval, so a cloned repository can't connect them without you.

Field marks

Loads	Cannot
When the session starts.	**Act on its own.**

Confused with Pl **Plugin**
A plugin can bring its own MCP server. You add or remove that server along with the plugin.

Plugin

Pl
04

Identify by A folder with `.claude-plugin/plugin.json`.

What **A plugin bundles skills, hooks and MCP servers.** Most come from a marketplace. An Accidental Guide to: Plugins, Connectors, and Marketplaces explains plugins.

So what **An enabled plugin is part of every session and runs as you.** Check what a plugin adds before you install it.

Field marks

Loads	Cannot
At startup, if turned on.	**Install for teammates.**

Confused with Sk **Skill**
A skill is one folder you can copy in yourself. A plugin is a package that can carry many skills and other parts.

Settings

Se
05

Identify by `settings.json`, `settings.local.json` or `managed-settings.json`.

What Settings are JSON files that decide which commands run without asking, which hooks run and which plugins are on.

So what **Permission rules allow, ask about or deny tool calls.** Hooks run commands at set points, such as before a tool call.

Field marks

Loads
At startup and on save.

Cannot
Override your organization.

Confused with

In **Instructions file**
Claude Code enforces settings. The model reads an instructions file as guidance.

Where do the files live?

Most setup sits in two `.claude/` folders, one in your home folder and one in the project.

User: you, in every project

In **Instructions**	`~/.claude/CLAUDE.md`
Sk **Skills**	`~/.claude/skills/`
Mc **MCP servers**	`~/.claude.json`
Pl **Plugins**	`~/.claude/plugins/`, turned on in `~/.claude/settings.json`
Se **Settings**	`~/.claude/settings.json`

Project: shared with the team, and local: only you

In **Instructions**	`CLAUDE.md` or `.claude/CLAUDE.md`, and `CLAUDE.local.md` for local
Sk **Skills**	`.claude/skills/`
Mc **MCP servers**	`.mcp.json`, and `~/.claude.json` for local
Pl **Plugins**	turned on in the project's settings files
Se **Settings**	`.claude/settings.json`, and `.claude/settings.local.json` for local

Managed: set by your organization

Folder	`/Library/Application Support/ClaudeCode/` on macOS, `/etc/claude-code/` on Linux, `C:\Program Files\ClaudeCode\` on Windows
In **Instructions**	`CLAUDE.md` in that folder
Sk **Skills**	`.claude/skills/` in that folder
Mc **MCP servers**	`managed-mcp.json` in that folder
Se **Settings**	`managed-settings.json` in that folder, or a policy from MDM or the admin console

Tip. `/status` shows which settings files loaded, `/context` the instructions files, `/mcp` the servers and `/plugin` the plugins.

Which one wins?

Some kinds add up across places. For others, one place wins.

Thing	When it's set in more than one place
Se **Settings**	Highest first: managed, the `--settings` flag, local, project, user. Lists such as permission rules combine.
Permission rules	Deny is checked first, then ask, then allow, so a deny anywhere blocks the call.
Hooks	All of them run.
In **Instructions**	All of them load. The file closest to where you started is read last.
Sk **Skills**	Same name: managed, then user, then project. Plugin skills carry the plugin's name, so they don't clash.
Mc **MCP servers**	Same name: local, then project, then user, then plugins.
Pl **Plugins**	Local, then project, then user.

Run `/status` to see which managed settings apply.

The same things in other tools

Each of these tools has its own guide.

Guide	What to look for there
An Accidental Guide to: Claude Desktop	Most setup is in the app. Local MCP servers go in `claude_desktop_config.json`.
An Accidental Guide to: GitHub Copilot CLI	`AGENTS.md` or `.github/copilot-instructions.md`, `.github/skills/`, and `~/.copilot/` for settings and MCP servers.
An Accidental Guide to: OpenAI Codex	`AGENTS.md`, `.agents/skills/`, and `~/.codex/config.toml` for settings and MCP servers.
An Accidental Guide to: VS Code and Visual Studio	`.github/copilot-instructions.md`, `.github/skills/` in VS Code, and a `.mcp.json` file for MCP servers.

Codex, GitHub Copilot CLI and VS Code read `AGENTS.md`, and Claude Code reads it when there's no `CLAUDE.md`. GitHub Copilot CLI and VS Code also read `.claude/skills/`.

Sources

Checked in October 2026. Claude Code changes often, so check the current docs before you rely on a path.

Claude Code: Settings
`code.claude.com/docs/en/settings`

Claude Code: Managed settings
`code.claude.com/docs/en/managed-settings`

Claude Code: Memory
`code.claude.com/docs/en/memory`

Claude Code: The .claude directory
`code.claude.com/docs/en/claude-directory`

Claude Code: Skills
`code.claude.com/docs/en/skills`

Claude Code: MCP
`code.claude.com/docs/en/mcp`

Claude Code: Install plugins
`code.claude.com/docs/en/plugins/install`

Claude Code: Permissions
`code.claude.com/docs/en/permissions`

Claude Code: Hooks
`code.claude.com/docs/en/hooks`

MCP: Connect to local servers
`modelcontextprotocol.io/docs/develop/connect-local-servers`

GitHub: Copilot CLI configuration
`docs.github.com/en/copilot/reference/copilot-cli-reference/cli-config-dir-reference`

OpenAI: Codex configuration
`learn.chatgpt.com/docs/config-file/config-basic`

VS Code: Agent skills
`code.visualstudio.com/docs/copilot/customization/agent-skills`



An Accidental Guide to…

Claude Code

A pocket-sized guide to where Claude Code keeps its setup, as of October 2026.
Written and reviewed by a person. AI does the grunt work. Set in TeX Gyre Heros and Newsreader.
Leave a star: `accidental.info/star`

My friend has a lot of questions.

---

oops@accidental.info · CC BY 4.0 · October 2026