SIDDHIO.AI · TOOLKIT

Siddhio.ai

5 Tools to Get More Out of Claude Code

MCP servers, skills, hooks, subagents and plugins: what each one does, with a starter example you can copy.

siddhio.com · Free guide · Facts checked November 2026
Independent guide. Siddhio.ai is not affiliated with the projects described.

The five building blocks

Tool	What it adds	Where it lives
MCP servers	Access to outside tools, databases and APIs	Added with `claude mcp add`
Skills	Reusable instructions Claude loads when relevant, or you run as /commands	.claude/skills/<name>/SKILL.md
Hooks	Commands that run automatically at set moments, like after every edit	settings.json
Subagents	Specialised helpers that work in their own context window	.claude/agents/<name>.md
Plugins	Bundles of skills, agents, hooks and MCP servers you install in one go	/plugin

1. MCP servers: connect Claude Code to your tools

MCP servers give Claude Code access to tools, databases and APIs, so you can ask it to work with your issue tracker, docs or data. Add them from your terminal, not inside a Claude session. This example connects the hosted Claude Code documentation server:

```
claude mcp add --transport http claude-code-docs https://code.claude.com/docs/mcp
claude mcp get claude-code-docs        # see which scope it was added to
claude mcp remove claude-code-docs     # remove it again

# make it available in every project
claude mcp add --scope user --transport http \
  claude-code-docs https://code.claude.com/docs/mcp
```

- HTTP is the recommended transport for remote servers; SSE is deprecated.
- Adding a server with the **project** scope writes it to `.mcp.json` so your team gets it too.
- Only connect servers you trust. Servers that pull in outside content can expose you to prompt injection.

2. Skills: teach Claude your workflows

A skill is a folder with a `SKILL.md` file. The folder name becomes a slash command, and the description tells Claude when to load it on its own. Personal skills go in `~/.claude/skills/` (all projects); project skills go in `.claude/skills/`. Old `.claude/commands/` files still work the same way.

```
mkdir -p ~/.claude/skills/summarize-changes

# ~/.claude/skills/summarize-changes/SKILL.md
---
name: summarize-changes
description: Summarize the uncommitted changes in this repo. Use when asked
  what changed or before writing a commit message.
---
Run `git status` and `git diff`, then write a short summary grouped by
feature, followed by any risks or files that need a second look.
```

Run it with `/summarize-changes`, or just ask “what did I change?”. Skills follow the open Agent Skills standard, so they also work in other tools that support it.

3. Hooks: automate the boring steps

Hooks are commands Claude Code runs at fixed points, such as before a tool runs (`PreToolUse`) or after it succeeds (`PostToolUse`). They're defined in `~/.claude/settings.json` (you), `.claude/settings.json` (the project) or `.claude/settings.local.json` (you, in this project). This one formats every file Claude edits with Prettier:

```
{
  "hooks": {
    "PostToolUse": [
      {
        "matcher": "Edit|Write",
        "hooks": [
          {
            "type": "command",
            "command": "jq -r '.tool_input.file_path' | xargs npx prettier -w"
          }
        ]
      }
    ]
  }
}
```

- The hook receives details of the tool call as JSON on standard input, which is what `jq` reads here.
- The `/hooks` menu shows your hooks; edit the settings file (or ask Claude) to change them.
- Hooks run with your own permissions, so read any hook before you add it.

4. Subagents: delegate focused jobs

Subagents are specialised assistants with their own context window, which keeps long searches and reviews out of your main conversation. Claude Code ships with built-in ones such as Explore, Plan and general-purpose, and you can add your own as Markdown files in `.claude/agents/` (project) or `~/.claude/agents/` (all projects). You can also just ask Claude to create one for you.

```
# ~/.claude/agents/code-reviewer.md
---
name: code-reviewer
description: Reviews recent code changes for bugs, security issues and
  readability. Use proactively after code changes.
tools: Read, Grep, Glob
---
You are a careful senior reviewer. Look at the latest changes, list real
problems first (with file and line), then smaller suggestions. Be brief.
```

Then say “Have the code-reviewer subagent look at my recent changes”. Clear, action-oriented descriptions help Claude pick the right subagent by itself.

5. Plugins: install a whole setup at once

A plugin bundles skills, agents, hooks and MCP servers into one install. Anthropic's official marketplace, `claude-plugins-official`, is available automatically. Run `/plugin` in a session to browse it, or install directly:

```
/plugin install github@claude-plugins-official

# if the marketplace isn't found, add or refresh it
/plugin marketplace add anthropics/claude-plugins-official
/plugin marketplace update claude-plugins-official

# Anthropic's demo marketplace with example plugins
/plugin marketplace add anthropics/claude-code
```

- From your normal shell the same install is `claude plugin install <name>@<marketplace>`.
- Plugins can run hooks and MCP servers, so read what a plugin contains before you install it.

Where to start

1. Write one skill for a task you repeat every week.
2. Add one hook that formats or checks your code automatically.
3. Connect one MCP server for the tool you switch to most often.

Sources

- MCP quickstart: https://code.claude.com/docs/en/mcp-quickstart
- MCP reference: https://code.claude.com/docs/en/mcp
- Skills: https://code.claude.com/docs/en/skills
- Hooks guide: https://code.claude.com/docs/en/hooks-guide
- Hooks reference: https://code.claude.com/docs/en/hooks
- Subagents: https://code.claude.com/docs/en/sub-agents
- Discover and install plugins: https://code.claude.com/docs/en/discover-plugins
- Plugins overview: https://code.claude.com/docs/en/plugins/overview