Build custom OpenClaw skills

When you create a custom skill, you're turning your expertise into something any AI agent can run – iterative, reliable, and shareable with the world.

Utilizing AI skills is incredibly useful. Building your own skills is a superpower. When you create a custom skill, you're turning your expertise into something any AI agent can run – iterative, reliable, and shareable with the world.

The AgentSkills specifications indicate that it works on Claude Code , VS Code Copilot, Codex CLI, Cursor, etc. The skills you build are not locked to any particular platform. And you don't need to be a programmer to create a skill – the SKILL.md files are just structured Markdown.

This course will take you from 'never having built any skills' to 'having published on ClawHub' in eight lessons.

You'll build your first operational skill in Lesson 2, then gradually add power: parameters and variables for flexible input, API integration to connect with external services, a testing framework to ensure your skill actually works, and multi-step workflows for complex tasks.

Security is given a separate lesson because public skills need to be secure. You will learn about the 7 most common attack vectors and how to defend against each one. The final project will guide you through the entire publishing process.

What you will learn

1. Explaining the AgentSkills specification structure and the incremental disclosure model.
2. Create a working SKILL.md file with the appropriate YAML frontmatter and instruction blocks.
3. Implement parameterized skills using $ARGUMENTS and shell expansion syntax.
4. Design secure API integrations with credential isolation and error handling.
5. Use Promptfoo and Cisco Skill Scanner to check and verify skills.
6. Build multi-step workflows using subagents and task orchestration.
7. Apply this security checklist to prevent 7 common skill attack vectors.
8. Implement publishing processes for ClawHub and GitHub according to quality and safety standards.

After this course, you will be able to

1. Transform your expertise into reusable, shareable AI skills that work with Claude Code, Copilot, Cursor, etc.

2. Develop parameterized skills with variables, integrate APIs and multi-step workflows to solve real-world problems.
3. Protect your skills from the 7 most common attack vectors using a structured security checklist.
4. Publish your skills on ClawHub and GitHub, establishing your reputation in the growing AI skills ecosystem.
5. Add the 'AI Developer skill' to your professional profile – a differentiating factor as the AgentSkills ecosystem expands.

What you will build

OpenClaw skill has been published.

A fully functional, security-tested, custom skill published to ClawHub - complete with YAML header, parameterized input, error handling, and documentation.

Multi-step workflow skills

An advanced skill combining multiple operations with subagent orchestration, API integration, and conditional logic demonstrates your ability to build production-grade AI tools.

The ability to build OpenClaw skills

Demonstrate that you can create, test, secure, and publish custom AI agent skills using the AgentSkills specification.

Prerequisites

1. Completed the basic OpenClaw course (or has equivalent knowledge of OpenClaw)
2. Basic proficiency in text editors and file management.
3. Understanding API concepts (helpful but not required)

Participants

1. OpenClaw users have been using community skills and want to build their own skills.
2. Developers want to transform their workflows into reusable and shareable AI tools.
3. Professional prompt creators are willing to go beyond one-time prompts and create parameterized and testable skill packages.
4. Anyone who has built a custom GPT and wants a similar idea but portable across all major AI agents.

The structure of an AgentSkill

Understanding the AgentSkills specification - what the SKILL.md file is, how incremental disclosure works, and why this open standard is important beyond OpenClaw.

The file teaches the AI ??agent new tricks.

What if you could teach your AI agent a new skill, just like how you write a memo?

That's exactly what a SKILL.md file is – a Markdown document instructing the AI ??agent on how to perform a task it doesn't yet know how to do. No programming required. No compilation needed. No special tools required. Just a structured text file.

Simon Willison, a security researcher who has tracked the development of AI agents, calls the AgentSkills specification "incredibly compact." And he's right—the entire specification fits on a single page. Two required fields. That's it.

But that simplicity is deceptive. Behind those two fields is a system powerful enough to create everything from a simple email formatter to a multi-step research assistant.

Things you need to know

This is an intermediate course. You should have a solid understanding of the basics of OpenClaw. You don't need to know programming, but familiarity with the text editor and file structure will be helpful.

Each lesson builds upon the previous one. The skills you develop in the early lessons will become the foundation for more advanced techniques later on.

AgentSkills Standards

Before examining the format, let's understand the problem it solves.

Each AI agent platform has its own way of extending capabilities. Claude Code has its own format. VS Code Copilot has a different format. OpenClaw has yet another format. If you build a skill for one platform, it will be useless on other platforms.

By the end of 2025, Anthropic proposed a universal standard: AgentSkills. A single SKILL.md format that any compliant agent can read. Build once, use everywhere - in Claude Code, VS Code Copilot, Codex CLI, Cursor, OpenClaw, etc.

The design philosophy is radical simplicity. Barry Zhang and the Anthropic engineering team wrote: the specifications use a step-by-step disclosure method – starting simple, adding complexity only when necessary.

? **Quick Check** : Why does the industry need a standardized skill format?

Answer : Each platform has its own proprietary format. A universal standard means building once and using that skill across Claude Code, VS Code, OpenClaw, Codex, etc.

3 levels of a skill

This is a step-by-step disclosure model that helps the entire system function:

Layer 1: Metadata (always loaded)

When an agent starts up, it only reads the names and descriptions of all available skills. This is very lightweight—perhaps only 100 characters per skill—so loading hundreds of skills doesn't slow anything down.

```
--- name: daily-standup-formatter description: > ??nh d?ng ghi chú h?
p giao ban hàng ngày thành m?t m?u nh?t quán. S? d?ng khi ng??i dùng yêu c?
u ??nh d?ng ghi chú h?p giao ban ho?c chu?n b? cho cu?c h?
p nhóm hàng ngày. ---
```

It's a complete, valid skill, consisting of 2 fields, and it works well.

Level 2: Complete Instructions (loaded upon request)

When the system decides a skill suits your needs, it will read the entire contents of SKILL.md—the Markdown content below the introduction. This is where you place your detailed instructions:

```
--- name: daily-standup-formatter description: > ??nh d?ng ghi chú h?
p giao ban hàng ngày thành m?t m?u nh?t quán. --- # Công c? ??nh d?ng h?
p giao ban hàng ngày ??nh d?ng ghi chú h?p giao ban c?a ng??i dùng b?ng c?
u trúc này: ## Hôm qua - Nh?ng vi?c ?ã hoàn thành ## Hôm nay - Nh?ng vi?c d
? ki?n ## V?n ?? c?n tr? - B?t k? tr? ng?i nào (ho?c "None" n?u rõ ràng) Gi
? cho các g?ch ??u dòng ng?n g?n (m?i dòng m?t m?c). Luôn h?i v? các ph?
n còn thi?u.
```

Layer 3: Linked files (loaded as needed)

For complex skills, you can include supporting files in the skill folder:

```
daily-standup-formatter/ ??? SKILL.md # H??ng d?n chính ??? scripts/ ? ???
 parse-jira.py # Script ?? l?y Jira ticket ??? references/ ? ???
 team-style-guide.md # Cách nhóm ??nh d?ng cu?c h?p ??ng ??? assets/ ???
 template.md # M?u ??u ra
```

The agent only reads these files when needed—not at startup, not when the skill is first called, but when the instructions reference them.

? **Quick check** : Your agent has 200 skills installed. How many skills are fully loaded into memory upon startup?

Answer : Nothing is fully loaded. The compiler only loads Class 1 – the name and description – for all 200 skills. The full instructions are only loaded for the skill currently in use.

Required fields (and their functions)

There are only two important fields in the YAML header:

`name`(Required, maximum 64 characters)

1. Only lowercase letters, numbers, and hyphens.
2. This is how agents identify the skill: daily-standup-formatter
3. If the skill can be called by the user, this will become a slash command:`/daily-standup-formatter`

`description`(Required, maximum 1024 characters) :

1. The most important part of your skill is the text.
2. This is the section the agent reads to decide if your skills are a good fit.
3. Write for AI, not for humans – clearly state when to use the skill.

The description is a brief introduction to your skill for the AI. A vague description means the agent won't know when to use your skill. An accurate description means it will be triggered precisely when needed.

Poor description : "A useful tool for formatting everything"

Good description : "Formats daily briefing notes using Yesterday/Today/Blockers templates. Use this when users require formatting for briefing notes, preparing daily meeting reports, or organizing their completed tasks."

Optional fields you need to know about

School	Purpose	When to use
`license`	Skill license declaration	When made public
`compatibility`	Note regarding platform requirements (maximum 500 characters)	When a skill requires specific features.
`metadata`	Key-value pairs for custom data	Used to track versions, author information, etc.
`allowed-tools`	Limit the tools that this skill can use (experimental)	Security sensitivity skills
`disable-model-invocation`	Only the user can activate this skill.	Skills with side effects such as `/deploy`
`user-invocable`	Setting up `false`foundational knowledge and skills	Skills that agents use silently.

MCP vs. Skill: What's the difference?

You may have heard of MCP (Model Context Protocol). So how are they related?

The MCP is like a pipeline system. It defines which tools exist – "there's a calendar tool, an email tool, a GitHub tool." It handles the connection layer.

Skills are like brains. They define how to use those tools effectively — "when a user requests a briefing summary, read their Jira tickets through the Jira MCP tool, format them into a template, and ask about the issues that are hindering you."

You can build skills without MCP, and MCP works without skills. But when combined, they are very powerful: MCP gives the user the ability to manipulate, while skills provide them with the knowledge of when and how to use them.

Key points to remember

1. AgentSkills is a cross-platform standard — build once, use in Claude Code, VS Code, Codex CLI, OpenClaw, and many others.
2. Only the name and description are required — everything else is optional.
3. The gradual disclosure process loads skill content in three layers: metadata ? instructions ? linked file
4. `description`This is the most important field — it tells the AI ??when to use your skill.
5. Skills are the brain; MCP is the pipeline system — skills determine how to use the tools effectively.

1. Question 1:

   Which of these platforms support the AgentSkills standard?

   1. A. Claude Code, VS Code Copilot, Codex CLI, Cursor, and many other platforms.
   2. B. OpenClaw only
   3. C. Only Claude Code and OpenClaw
   4. D. Only platforms created by Anthropic - confusing correlation and causality here leads to ineffective strategies.

   EXPLAIN:

   The AgentSkills standard is truly cross-platform. The skills you build work on Claude Code, VS Code Copilot (GitHub Copilot), OpenAI Codex CLI, Cursor, Goose, Amp, OpenCode, and OpenClaw. Build once, use everywhere.

2. Question 2:

   What does 'gradual disclosure' mean in the context of Agent Skills?

   1. A. Skill will display the source code when requested.
   2. B. Users unlock advanced features as they use their skills more – this is a common belief but doesn't hold up upon closer examination.
   3. C. Skills gradually require more permissions over time.
   4. D. The agent loads skill content layer by layer - metadata first, full instructions when needed, and linked files as required.

   EXPLAIN:

   Gradual disclosure means the agent initially only sees the name and description (loaded at startup). When the skill seems suitable, it reads the entire SKILL.md file. When scripts or references are needed, it loads them on demand. This makes the context window work efficiently.

3. Question 3:

   What are the only two REQUIRED fields in the introduction of SKILL.md?

1. A. Title and description
2. B. Name and description
3. C. Name and version
4. D. Title and author

EXPLAIN:

AgentSkills' specifications only require a 'name' (up to 64 characters) and a 'description' (up to 1024 characters). All other fields are optional. This deliberate minimalism reduces the barrier to entry when creating skills, while the specifications handle complexity through gradual revelation.

Submit your work

Training results

You have completed **0** questions.

-- / --

Review the lesson