

Module 07 — Skills and ClawHub



Teaching Your Agent New Abilities



Browse, inspect, install, and build skills

Navigation Chart

By the end of this module, you will be able to:


1. **Explain** what 🐟 skills are and how they work
2. **Browse and install** skills from 🦀 ClawHub (the skill marketplace)
3. **Evaluate** a skill for security before installing it
4. **Build** a simple custom skill from scratch
5. **Understand** the token impact of skills on your context window
6. **Manage** your installed skills (enable, disable, remove)

Ship's Logbook

Term	Definition
Skill	A plugin for 🦸 OpenClaw — markdown with YAML front matter that teaches your agent a new capability
ClawHub	The 🦸 community marketplace for OpenClaw skills (5,700+ available)
YAML front matter	The metadata section at the top of a skill file, between <code>---</code> markers
Token impact	The context window space a skill uses. More skills = less room for conversation

What Are Skills?

Without skills, your agent can chat, run commands, and read/write files.

With  skills, your agent can:

- Send and manage emails
- Create images with AI
- Interact with Google Workspace
- Post to Twitter/X
- Take and search notes in Obsidian
- Control smart home devices
- And thousands more

A skill is just a markdown file with special metadata. When enabled, its content is loaded into the agent's context -- giving your agent new instructions

Skill YAML Structure

Every 🐟 skill follows this format:

```
---
name: "email-assistant"
description: "Manages email: read, write, reply, search"
triggers:
  - "email"
  - "send email"
  - "check my inbox"
tools:
  - "himalaya"
---
```

- **name** — Unique identifier for the skill
- **description** — What the skill does
- **triggers** — Keywords that activate the skill

Skill Markdown Body

Below the YAML front matter, write the instructions:

Email Assistant

When the user asks about email, use Himalaya CLI:

1. ****Read emails****: `himalaya list`
2. ****Read specific****: `himalaya read <id>`
3. ****Send email****: `himalaya send --to <addr>`
4. ****Reply****: `himalaya reply <id>`

Rules

- Always confirm before sending
- Show a preview before sending
- Never auto-reply without user approval

When this skill is active and you say "check my inbox," your agent knows exactly what tool to use and how.

Browsing ClawHub

Over 5,700 skills available. Browse them:

```
openclaw skills browse      # Terminal
```

```
/skills browse              # TUI
```

Category	Examples
Communication	Email, Slack, SMS, Telegram enhancements
Productivity	Obsidian, Notion, task management, calendar
Development	GitHub, code review, deployment
Research	Web search, data analysis, academic papers
Social Media	Twitter/X, LinkedIn, Instagram posting
Creative	Image generation, writing, music

Installing a Skill: Step 1 — Find

```
openclaw skills search "obsidian"
```

Results:

1. obsidian-notes (v1.2.3) - Read, write, search vault
4/5 stars (423 installs)
2. obsidian-daily (v0.8.1) - Daily notes management
3/5 stars (156 installs)
3. obsidian-tasks (v2.0.0) - Task management in vault
5/5 stars (812 installs)

Look at **star rating** and **install count** as initial trust signals.

► Rough Waters: Skill Inspection

Do not install skills blindly. Read the content first:

```
openclaw skills inspect obsidian-notes
```

What to look for:

- Does it request more permissions than it needs?
- Does the markdown contain suspicious instructions?
- Does it run commands unrelated to its purpose?
- Is the author reputable? How many installs?

Red flags:

- Skills requesting elevated/admin access
- Instructions to send data to external servers

Installing a Skill: Steps 3-5

Step 3: Install (with version pinning)

```
openclaw skills install obsidian-notes --pin
```

Always use `--pin` to lock the version and detect integrity drift.

Step 4: Restart

```
openclaw service restart
```

Step 5: Test

Open the TUI and try:

```
Can you list the notes in my Obsidian vault?
```

VirusTotal Scanning

 ClawHub partners with VirusTotal to scan skills for malicious content.

What it catches:

- Known malware patterns
- Suspicious URLs
- Obvious data exfiltration attempts

What it does NOT catch:

- Subtle prompt injection instructions
- Novel attack patterns
- Social engineering embedded in instructions

VirusTotal is one layer. Your manual inspection is another layer. Neither alone is sufficient.

🚩 Rough Waters: The ClawHub Supply Chain Attack (2026)

- **1,184 malicious skills** found on 🦀 ClawHub
- One attacker uploaded **677 packages** alone
- Disguised as crypto bots, YouTube summarizers, wallet trackers
- Professional documentation hid malicious instructions in SKILL.md
- **#1 ranked skill** ("What Would Elon Do") had 9 vulnerabilities (Cisco scan)
- Payloads: **Atomic Stealer** (passwords, SSH keys, crypto wallets), **reverse shells**
- Rankings were **gamed** to reach #1

This is npm supply chain attacks -- except the package can think and has shell access.

Always inspect. Never trust install counts alone.

Building a Custom Skill

Step 1: Create the file

```
nano ~/.openclaw/workspace/skills/daily-standup.md
```

Step 2: Write the YAML + markdown

```
---
name: "daily-standup"
description: "Generates a daily standup report"
triggers:
  - "standup"
  - "daily standup"
  - "what did I do yesterday"
---
```

```
# Daily Standup Generator
```

```
When the user asks for a standup, generate:
```

Custom Skill: The Markdown Body

Format

Yesterday:

- [Check session history and memory]

Today:

- [Check user goals and scheduled tasks]

Blockers:

- [Any issues mentioned recently]

Rules

- Keep each section to 3-5 bullet points
- Pull from actual history, don't make things up
- If not enough info, ask the user
- Format as clean markdown

Save (Ctrl + O , Enter, Ctrl + X), restart (openclaw service restart), and test ("Give

Custom Skill Ideas

Build 🐟 skills that match YOUR workflow:

- **Research template** — how you want research formatted (executive summary + findings + sources)
- **Meeting notes** — structured format for meeting summaries
- **Decision framework** — pros/cons analysis template
- **Weekly review** — questions for your weekly reflection
- **Client onboarding** — checklist for new freelance clients

The key principle: if you can explain the process to a human, you can write it as a skill.

Skill Best Practices

Write Better Descriptions

Include **"Use when"** AND **"Don't use when"** in descriptions:

```
description: "Deep research. Use when user asks for analysis.  
Don't use when user just wants a quick factual answer."
```

Put Templates Inside Skills

Templates cost **zero tokens when inactive** -- they only load when triggered. Build rich, detailed skills without context bloat.

Self-Improving Skills

Add a "Lessons" section so your agent records what worked and failed. The agent never makes the same mistake twice.

The "Twice = Skill" Rule & Channels as Departments

If You Do Something More Than Twice, Make a 🐟 Skill for It

- The first time is exploration. The second time is a pattern. The **third time is waste**.
- Turn any repeated workflow into a skill and never explain the process again.

Channels as Departments

Map each messaging channel to a specific skill:

Channel	Skill	Agent Behavior
#x-scan	X/Twitter skill	Scan threads, summarize posts, draft replies
#finances	Finance skill	Track expenses, check markets, generate reports
#writing	Writing skill	Draft, edit, format content
#research	Research skill	Deep search, summarize sources, save findings

- The agent knows what **"mode" to be in** based on the channel it receives a message from

Guardrail Skills

Not all skills add features. Some skills **prevent disasters**.

The **anti-loop.md** Pattern

A tiny skill (~50 tokens) that saves you from runaway cost spirals:

```
---  
name: "anti-loop"  
description: "Prevents the agent from looping on errors"  
triggers: ["error", "failed", "retry"]  
---  
  
If you see the same error twice, STOP and ask the user.  
Check USER.md before asking questions the user  
may have already answered.
```

Why This Matters

- **Loops on Opus can burn \$5-20 in minutes** -- the agent retries the same failing approach

Token Impact of 🐟 Skills

Every active skill takes up context window space.

When you start a conversation, 🦏 **OpenClaw loads:**

- Your core files (identity, soul, user, 🔥 memory, agents, etc.)
- All active skill instructions
- The conversation history

The problem: 50 skills x 500 tokens each = 25,000 tokens used before you say a word. That means less room for conversation, more /compact , and higher costs.

The solution:

- Only enable skills you actively use
- Keep custom skills concise
- Review and remove periodically

Managing Installed Skills

```
# List all installed skills
openclaw skills list

# Disable (keeps installed, doesn't load)
openclaw skills disable obsidian-notes

# Enable a previously disabled skill
openclaw skills enable obsidian-notes

# Remove entirely
openclaw skills remove obsidian-notes
```

Disable vs. Remove: *Disable keeps the skill for later. Remove deletes it. Prefer disabling unless you are sure you will never need it again.*

► Shoals and Sandbars

Mistake	Fix
Installing skills without reading them	Always use <code>openclaw skills inspect</code> first
Installing too many skills at once	Start with 2-3, add more as needed
Not restarting after installing	Run <code>openclaw service restart</code>
Skill does not seem to work	Check trigger words in the YAML front matter
Custom skill has vague instructions	Be specific in your markdown body
Leaving unused skills enabled	Disable or remove to save tokens

Hands on Deck: Install One, Build One

Part 1: Install from ClawHub (10 min)

1. Browse: `openclaw skills browse`
2. Search for something relevant to you
3. Inspect: `openclaw skills inspect [skill-name]`
4. Check for security red flags
5. Install: `openclaw skills install [skill-name]`
6. Restart: `openclaw service restart`
7. Test it in the TUI

Part 2: Build a Custom Skill (15 min)

1. Create: `nano ~/.openclaw/workspace/skills/my-skill.md`
2. Write the YAML front matter and markdown instructions

Hands on Deck (continued)

Part 3: Evaluate (5 min)

- Does the installed skill work as expected?
- Does your custom skill produce the output you wanted?
- What would you improve about either skill?
- How many tokens do your active skills consume? (Check with `/status`)

Treasure Chest

1. **Skills are markdown files with YAML metadata** — they teach your agent new capabilities through instructions
2. **ClawHub has 5,700+ skills** — browse before building from scratch
3. **Always inspect before installing** — read the full content and check for security issues
4. **Pin your skill versions** — use `--pin` to lock versions and detect integrity drift
5. **Token impact matters** — every active skill uses context window space; only keep what you need
6. **Building custom skills is easy** — if you can write instructions for a human, you can write a skill
7. **VirusTotal scanning helps but is not sufficient** — your manual review is the last line of defense
8. **Disable unused skills** — keeps your context lean and your costs down



Next Port of Call

Module 08: Cron Jobs and Heartbeats

Skills are reactive -- they work when you ask. In Module 08, we make your agent proactive. Morning briefings, periodic check-ins, monitoring tasks -- your agent will work for you even when you are not asking.