Cẩm nang đầy đủ xây dựng Skill cho Claude

Claude

Phiên bản tiếng Việt · Biên dịch & Việt hóa bởi Phong Hồ

Mục lục

Giới thiệu

Một **skill** là một bộ hướng dẫn, được đóng gói gọn trong một thư mục, dạy Claude cách xử lý những công việc hoặc quy trình cụ thể. Đây là một trong những cách mạnh mẽ nhất để bạn điều chỉnh Claude theo nhu cầu riêng. Thay vì lặp lại sở thích, quy trình và kiến thức chuyên môn của mình trong từng cuộc trò chuyện, bạn chỉ cần dạy Claude một lần rồi hưởng lợi mãi về sau.

Skill phát huy sức mạnh khi bạn có những quy trình lặp đi lặp lại: dựng giao diện frontend từ bản đặc tả, nghiên cứu theo một phương pháp thống nhất, soạn tài liệu đúng chuẩn trình bày của nhóm, hay điều phối các quy trình nhiều bước. Skill ăn ý với các năng lực sẵn có của Claude như chạy code và tạo tài liệu. Còn với người làm tích hợp MCP, skill là một lớp nữa giúp biến quyền truy cập công cụ thô thành những quy trình đáng tin cậy và đã được tối ưu.

Cẩm nang này bao quát mọi thứ bạn cần để xây một skill hiệu quả, từ khâu lên kế hoạch, dựng cấu trúc, đến kiểm thử và phân phối. Dù bạn làm skill cho riêng mình, cho nhóm, hay cho cộng đồng, bạn đều sẽ tìm thấy những cách làm thiết thực và ví dụ thực tế xuyên suốt.

Bạn sẽ học được gì:

- Yêu cầu kỹ thuật và cách làm tốt nhất khi dựng cấu trúc skill
- Các mẫu cho skill độc lập và cho quy trình được tăng cường bằng MCP
- Những mẫu mà chúng tôi thấy chạy tốt trong nhiều tình huống khác nhau
- Cách kiểm thử, cải tiến và phân phối skill của bạn

Cẩm nang dành cho ai:

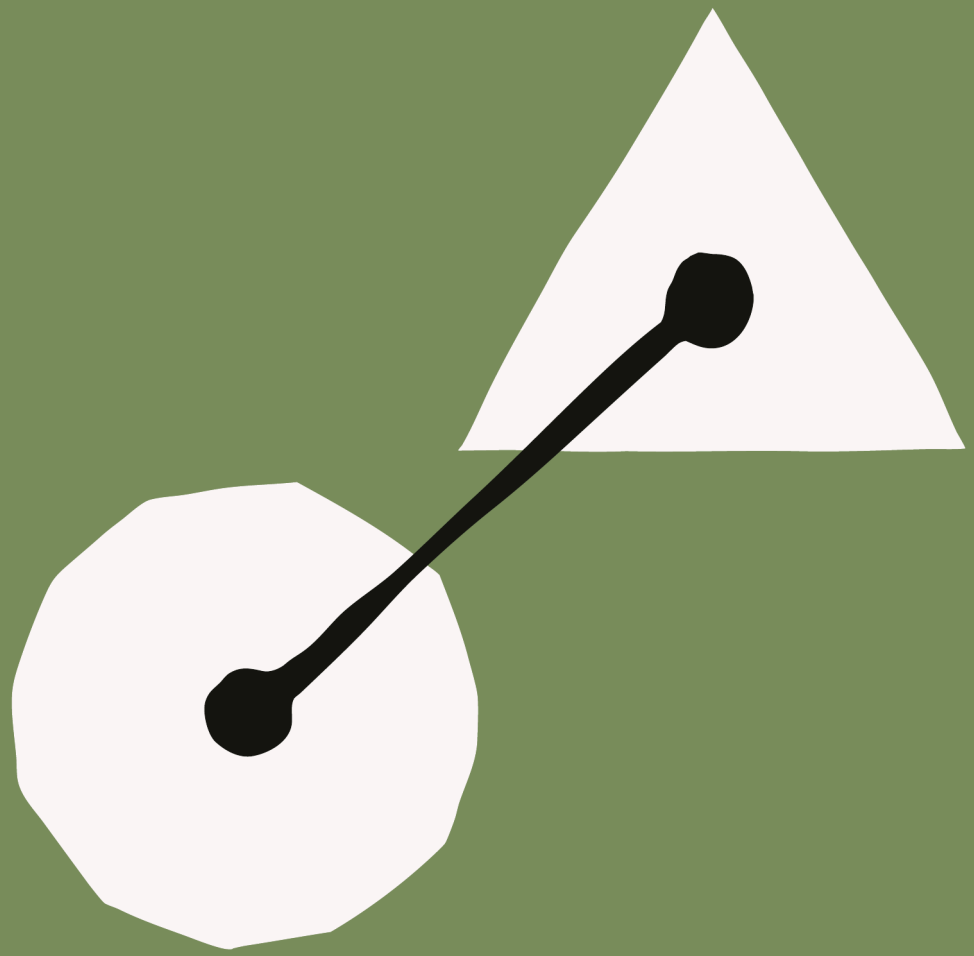
- Lập trình viên muốn Claude bám theo quy trình cụ thể một cách nhất quán
- Người dùng thành thạo muốn Claude làm đúng theo quy trình của mình
- Các nhóm muốn chuẩn hóa cách Claude làm việc trong cả tổ chức

Hai hướng đi qua cẩm nang này

Bạn đang làm skill độc lập? Hãy tập trung vào phần Nền tảng, Lên kế hoạch và thiết kế, cùng nhóm 1 và 2. Bạn đang tăng cường cho một tích hợp MCP? Phần "Skill + MCP" và nhóm 3 là dành cho bạn. Cả hai hướng đều dùng chung bộ yêu cầu kỹ thuật, chỉ là bạn chọn phần nào hợp với mình.

Bạn sẽ đạt được gì sau cẩm nang: Đến cuối, bạn có thể dựng xong một skill chạy được chỉ trong một lần ngồi làm. Dự kiến khoảng 15 đến 30 phút để dựng và kiểm thử skill đầu tiên với skill-creator.

Bắt đầu thôi.



Chương 1

Nền tảng

Nền tảng

Skill là gì?

Một skill là một thư mục gồm:

- `SKILL.md` (bắt buộc): phần hướng dẫn viết bằng Markdown, kèm phần đầu YAML (frontmatter)
- `scripts/` (tùy chọn): mã chạy được (Python, Bash, v.v.)
- `references/` (tùy chọn): tài liệu được nạp khi cần
- `assets/` (tùy chọn): mẫu, font, icon dùng trong kết quả

Nguyên tắc thiết kế cốt lõi

Tiết lộ theo từng lớp (Progressive Disclosure)

Skill dùng cơ chế ba lớp:

- **Lớp một (phần đầu YAML):** luôn được nạp vào system prompt của Claude. Chỉ cung cấp vừa đủ thông tin để Claude biết khi nào nên dùng mỗi skill, mà không phải nạp toàn bộ vào ngữ cảnh.
- **Lớp hai (thân SKILL.md):** được nạp khi Claude thấy skill có liên quan đến công việc hiện tại. Chứa toàn bộ hướng dẫn chi tiết.
- **Lớp ba (các file liên kết):** những file bổ sung nằm trong thư mục skill, Claude tự tìm và mở khi cần.

Cơ chế tiết lộ theo từng lớp này giúp tiết kiệm token tối đa mà vẫn giữ được chuyên môn chuyên sâu.

Khả năng kết hợp

Claude có thể nạp nhiều skill cùng lúc. Skill của bạn nên hoạt động ăn ý bên cạnh các skill khác, đừng mặc định rằng nó là năng lực duy nhất đang có.

Dùng được trên mọi nền tảng

Skill chạy y như nhau trên Claude.ai, Claude Code và API. Bạn tạo skill một lần là dùng được ở mọi nơi mà không phải sửa, miễn là môi trường đó hỗ trợ những thứ skill cần.

Dành cho người làm MCP: Skill + Connector

💡 Bạn đang làm skill độc lập, không dùng MCP? Hãy nhảy thẳng tới phần Lên kế hoạch và thiết kế, lúc nào cần thì quay lại đây sau.

Nếu đã có sẵn **một MCP server chạy được**, bạn đã làm xong phần khó nhất. Skill là lớp kiến thức nằm bên trên, ghi lại những quy trình và cách làm tốt mà bạn đã biết, để Claude áp dụng đều đặn.

Ví von với căn bếp

MCP là căn bếp chuyên nghiệp: cho bạn dụng cụ, nguyên liệu và thiết bị.

Skill là công thức nấu ăn: chỉ dẫn từng bước để làm ra món có giá trị.

Kết hợp lại, chúng giúp người dùng hoàn thành những việc phức tạp mà không phải tự mò từng bước.

Hai bên bổ trợ nhau thế nào:

MCP (kết nối)	Skill (kiến thức)
Kết nối Claude với dịch vụ của bạn (Notion, Asana, Linear, v.v.)	Dạy Claude cách dùng dịch vụ của bạn cho hiệu quả
Cấp quyền truy cập dữ liệu thời gian thực và gọi công cụ	Ghi lại các quy trình và cách làm tốt
Claude làm được gì	Claude nên làm việc đó ra sao

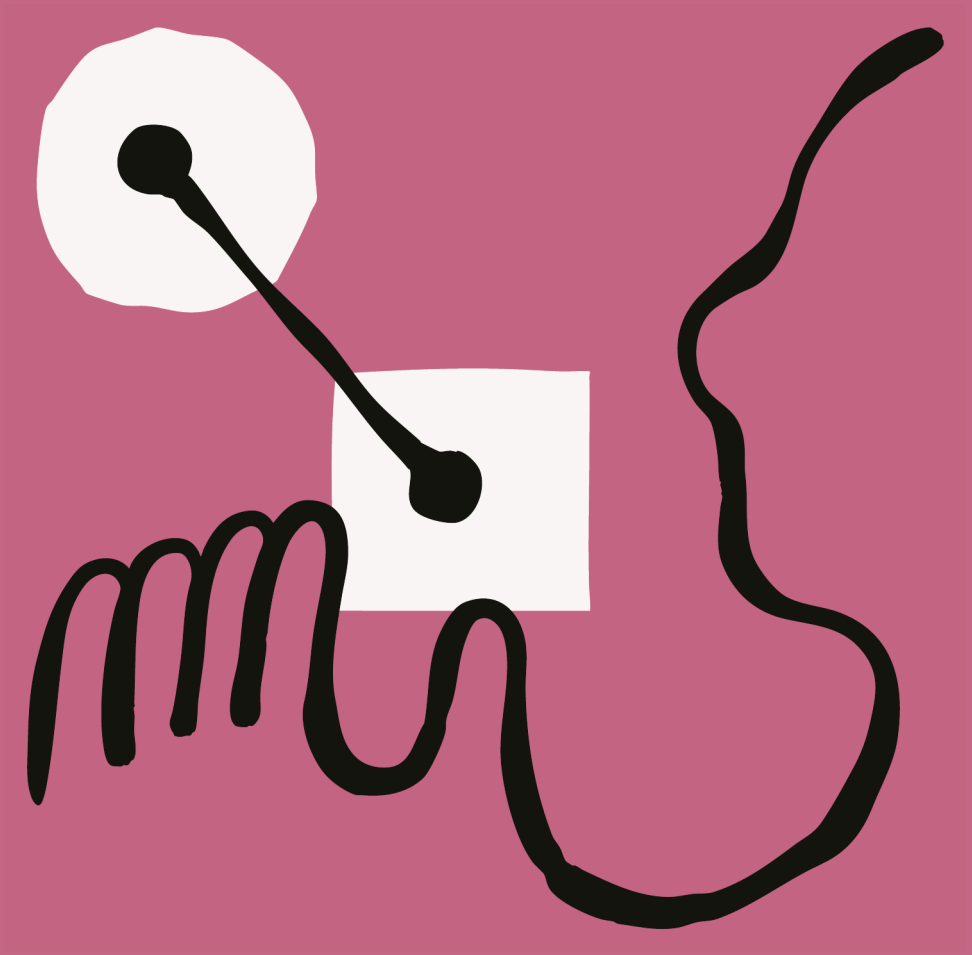
Vì sao điều này quan trọng với người dùng MCP của bạn

Khi chưa có skill:

- Người dùng kết nối MCP của bạn nhưng không biết bước tiếp theo làm gì
- Tràn ngập câu hỏi hỗ trợ kiểu "làm X với tích hợp của bạn thế nào"
- Mỗi cuộc trò chuyện đều bắt đầu lại từ con số không
- Kết quả không nhất quán vì mỗi người ra lệnh một kiểu
- Người dùng trách connector của bạn, trong khi vấn đề thật ra là thiếu hướng dẫn quy trình

Khi có skill:

- Các quy trình dựng sẵn tự kích hoạt đúng lúc cần
- Dùng công cụ nhất quán, đáng tin cậy
- Cách làm tốt được cài sẵn trong mọi lần tương tác
- Người mới đỡ mất công làm quen với tích hợp của bạn



Chương 2

Lên kế hoạch và thiết kế

Chương 2

Lên kế hoạch và thiết kế

Bắt đầu từ tình huống sử dụng

Trước khi viết bất kỳ dòng code nào, hãy xác định 2 đến 3 tình huống sử dụng cụ thể mà skill của bạn cần đáp ứng.

Một tình huống được định nghĩa tốt:

```
Tình huống: Lập kế hoạch sprint cho dự án
Kích hoạt khi: người dùng nói "giúp tôi lên kế
  hoạch sprint này" hoặc "tạo các nhiệm vụ sprint"
Các bước:
  1. Lấy trạng thái dự án hiện tại từ Linear (qua MCP)
  2. Phân tích tốc độ và năng lực của nhóm
  3. Đề xuất thứ tự ưu tiên công việc
  4. Tạo nhiệm vụ trong Linear kèm nhãn và ước lượng
Kết quả: sprint được lên kế hoạch đầy đủ, đã tạo nhiệm vụ
```

Tự hỏi mình:

- Người dùng muốn làm xong việc gì?
- Việc đó cần những quy trình nhiều bước nào?
- Cần công cụ gì (có sẵn hay qua MCP)?
- Cần cài sẵn kiến thức chuyên môn hay cách làm tốt nào?

Các nhóm tình huống sử dụng phổ biến

Tại Anthropic, chúng tôi thấy có ba nhóm tình huống sử dụng phổ biến:

Nhóm 1: Tạo tài liệu và sản phẩm

Dùng cho: tạo ra kết quả nhất quán, chất lượng cao như tài liệu, bản trình bày, ứng dụng, thiết kế, code, v.v.

Ví dụ thực tế: skill **frontend-design** (xem thêm các **skill cho docx, pptx, xlsx** và **ppt**)

"Tạo giao diện frontend đặc sắc, đạt chuẩn production với chất lượng thiết kế cao. Dùng khi cần dựng component web, trang, artifact, poster hoặc ứng dụng."

Kỹ thuật chính:

- Cài sẵn chuẩn trình bày và bộ nhận diện thương hiệu
- Có cấu trúc mẫu để kết quả luôn nhất quán
- Có danh sách kiểm tra chất lượng trước khi chốt
- Không cần công cụ ngoài, chỉ dùng năng lực sẵn có của Claude

Nhóm 2: Tự động hóa quy trình

Dùng cho: các quy trình nhiều bước cần một phương pháp nhất quán, gồm cả việc phối hợp giữa nhiều MCP server.

Ví dụ thực tế: skill **skill-creator**

"Hướng dẫn tương tác để tạo skill mới. Dẫn người dùng đi qua các bước xác định tình huống sử dụng, tạo phần đầu YAML, viết hướng dẫn và kiểm tra."

Kỹ thuật chính:

- Quy trình theo từng bước, có chốt kiểm tra ở mỗi bước
- Mẫu cho các cấu trúc thường gặp
- Tích hợp sẵn việc rà soát và gợi ý cải tiến
- Vòng lặp tinh chỉnh dần

Nhóm 3: Tăng cường cho MCP

Dùng cho: bổ sung hướng dẫn quy trình để khai thác tốt hơn quyền truy cập công cụ mà một MCP server đem lại.

Ví dụ thực tế: skill **sentry-code-review** (của Sentry)

"Tự động phân tích và sửa các lỗi phát hiện được trong Pull Request trên GitHub, dựa vào dữ liệu giám sát lỗi của Sentry qua MCP server của họ."

Kỹ thuật chính:

- Điều phối nhiều lệnh gọi MCP theo trình tự
- Cài sẵn chuyên môn theo lĩnh vực
- Cung cấp ngữ cảnh mà lẽ ra người dùng phải tự nêu
- Xử lý các lỗi MCP thường gặp

Xác định tiêu chí thành công

Làm sao bạn biết skill của mình đang chạy tốt?

Đây là những mục tiêu mang tính định hướng, là mức tham chiếu tương đối chứ không phải ngưỡng chính xác. Hãy cố làm cho chặt chẽ, nhưng cũng chấp nhận rằng vẫn còn một phần đánh giá theo cảm tính. Chúng tôi đang tích cực phát triển hướng dẫn và công cụ đo lường bài bản hơn.

Chỉ số định lượng:

- Skill kích hoạt đúng ở 90% các câu hỏi liên quan
  - Cách đo: chạy thử 10 đến 20 câu hỏi lẽ ra phải kích hoạt skill. Đếm số lần nó tự nạp so với số lần phải gọi tay.
- Hoàn thành quy trình trong X lần gọi công cụ
  - Cách đo: so cùng một việc khi bật và khi tắt skill. Đếm số lần gọi công cụ và tổng token tiêu thụ.
- Không có lần gọi API nào thất bại trong mỗi quy trình
  - Cách đo: theo dõi log của MCP server trong lúc chạy thử. Đếm tỉ lệ thử lại và mã lỗi.

Chỉ số định tính:

- Người dùng không phải nhắc Claude về bước kế tiếp
  - Cách đánh giá: trong lúc thử, để ý xem bạn phải chỉnh hướng hay nói rõ lại bao nhiêu lần. Hỏi thêm ý kiến người dùng thử nghiệm.
- Quy trình chạy xong mà không cần người dùng sửa
  - Cách đánh giá: chạy cùng một yêu cầu 3 đến 5 lần. So kết quả về độ nhất quán cấu trúc và chất lượng.
- Kết quả ổn định qua các phiên làm việc
  - Cách đánh giá: một người dùng mới có làm xong việc ngay lần đầu với chỉ dẫn tối thiểu không?

Yêu cầu kỹ thuật

Cấu trúc thư mục

```
your-skill-name/
├── SKILL.md                # Bắt buộc - file skill chính
├── scripts/                # Tùy chọn - mã chạy được
│   ├── process_data.py # Ví dụ
│   └── validate.sh # Ví dụ
├── references/             # Tùy chọn - tài liệu
│   ├── api-guide.md # Ví dụ
│   └── examples/ # Ví dụ
└── assets/                 # Tùy chọn - mẫu, v.v.
    └── report-template.md # Ví dụ
```

Các quy tắc quan trọng

Đặt tên SKILL.md:

- Phải đúng chính xác là `SKILL.md` (phân biệt hoa thường)
- Không chấp nhận biến thể (SKILL.MD, skill.md, v.v.)

Đặt tên thư mục skill:

- Dùng kebab-case: `notion-project-setup` ✓
- Không khoảng trắng: `Notion Project Setup` ✗
- Không gạch dưới: `notion_project_setup` ✗
- Không viết hoa: `NotionProjectSetup` ✗

Không dùng README.md:

- Đừng để README.md bên trong thư mục skill
- Mọi tài liệu nằm trong SKILL.md hoặc references/
- Lưu ý: khi phát hành qua GitHub, bạn vẫn nên có một README ở cấp repo cho người đọc, xem phần Phân phối và chia sẻ.

Phần đầu YAML: phần quan trọng nhất

Phần đầu YAML (frontmatter) là nơi Claude quyết định có nạp skill của bạn hay không. Hãy làm cho thật chuẩn.

Định dạng tối thiểu bắt buộc

```yaml
---
name: your-skill-name
description: Mô tả skill làm gì. Dùng khi người
  dùng yêu cầu [các cụm từ cụ thể].
---
```

Chỉ cần vậy là bắt đầu được.

Yêu cầu cho từng trường

`name` (bắt buộc):

- Chỉ dùng kebab-case
- Không khoảng trắng hay chữ hoa
- Nên trùng với tên thư mục

`description` (bắt buộc):

- BẮT BUỘC có CẢ HAI:
  - Skill làm gì
  - Khi nào dùng (điều kiện kích hoạt)
- Dưới 1024 ký tự
- Không có thẻ XML (< hoặc >)
- Nêu rõ những câu người dùng có thể nói
- Nhắc loại file nếu có liên quan

`license` (tùy chọn):

- Dùng nếu phát hành skill dạng mã nguồn mở
- Phổ biến: MIT, Apache-2.0

`compatibility` (tùy chọn)

- 1 đến 500 ký tự
- Cho biết yêu cầu môi trường: ví dụ sản phẩm dự kiến chạy, gói hệ thống cần có, nhu cầu truy cập mạng, v.v.

`metadata` (tùy chọn):

- Bất kỳ cặp khóa-giá trị tùy chỉnh nào
- Gợi ý: author, version, mcp-server
- Ví dụ:

```
metadata:
  author: ProjectHub
  version: 1.0.0
  mcp-server: projecthub
```

Giới hạn bảo mật

Cấm trong phần đầu YAML:

- Dấu ngoặc nhọn XML (< >)
- Skill có chữ "claude" hoặc "anthropic" trong tên (đã được giữ riêng)

Vì sao: phần đầu YAML xuất hiện trong system prompt của Claude. Nội dung độc hại có thể lợi dụng để chèn lệnh.

Viết skill cho hiệu quả

Trường description

Theo **blog kỹ thuật của Anthropic**: phần thông tin này "chỉ cung cấp vừa đủ để Claude biết khi nào nên dùng mỗi skill, mà không phải nạp toàn bộ vào ngữ cảnh." Đây chính là lớp đầu tiên của cơ chế tiết lộ theo từng lớp.

Cấu trúc:

```
[Skill làm gì] + [Khi nào dùng] + [Năng lực chính]
```

Vài ví dụ description tốt:

```
# Tốt - cụ thể và dùng được ngay
description: Phân tích file thiết kế Figma và tạo
tài liệu bàn giao cho lập trình viên. Dùng khi
người dùng tải lên file .fig, hoặc yêu cầu "thông
số thiết kế", "tài liệu component", "bàn giao từ
thiết kế sang code".

# Tốt - có cụm từ kích hoạt
description: Quản lý quy trình dự án trên Linear,
gồm lập kế hoạch sprint, tạo nhiệm vụ và theo dõi
trạng thái. Dùng khi người dùng nhắc "sprint",
"nhiệm vụ Linear", "lập kế hoạch dự án", hoặc yêu
cầu "tạo ticket".

# Tốt - nêu rõ giá trị mang lại
description: Quy trình tiếp nhận khách hàng trọn
gói cho PayFlow. Lo việc tạo tài khoản, thiết lập
thanh toán và quản lý gói đăng ký.
```

Vài ví dụ description tệ:

```
# Quá mơ hồ
description: Giúp việc với các dự án.

# Thiếu điều kiện kích hoạt
description: Tạo hệ thống tài liệu nhiều trang
phức tạp.

# Quá kỹ thuật, không có câu kích hoạt của người dùng
description: Hiện thực mô hình thực thể Project với
các quan hệ phân cấp.
```

Viết phần hướng dẫn chính

Sau phần đầu YAML, hãy viết phần hướng dẫn thực tế bằng Markdown.

Cấu trúc gợi ý:

Điều chỉnh mẫu này cho skill của bạn. Thay phần trong ngoặc [] bằng nội dung cụ thể của bạn.

```
---
name: your-skill
description: [...]
---

# Tên Skill Của Bạn

## Hướng dẫn
### Bước 1: [Bước lớn đầu tiên]
Giải thích rõ điều gì sẽ diễn ra.
```

Ví dụ:

```
python scripts/fetch_data.py --project-id PROJECT_ID
Kết quả mong đợi: [mô tả thế nào là thành công]
```

(Thêm bước nếu cần)

Ví dụ minh họa

Ví dụ 1: [tình huống thường gặp]

Người dùng nói: "Thiết lập một chiến dịch marketing mới"

Hành động:

- Lấy các chiến dịch hiện có qua MCP
- Tạo chiến dịch mới với tham số được cung cấp

Kết quả: chiến dịch được tạo, kèm link xác nhận

(Thêm ví dụ nếu cần)

Xử lý sự cố

Lỗi: [thông báo lỗi thường gặp]

Nguyên nhân: [vì sao xảy ra]

Cách khắc phục: [sửa thế nào]

(Thêm trường hợp lỗi nếu cần)

Cách viết hướng dẫn tốt

Cụ thể và làm được ngay

✓ **Tốt:**

```
Chạy `python scripts/validate.py --input
{filename}` để kiểm tra định dạng dữ liệu.
Nếu kiểm tra không đạt, các lỗi thường gặp gồm:
- Thiếu trường bắt buộc (thêm vào file CSV)
- Sai định dạng ngày (dùng YYYY-MM-DD)
```

✗ **Tệ:**

```
Kiểm tra dữ liệu trước khi tiếp tục.
```

Có phần xử lý lỗi

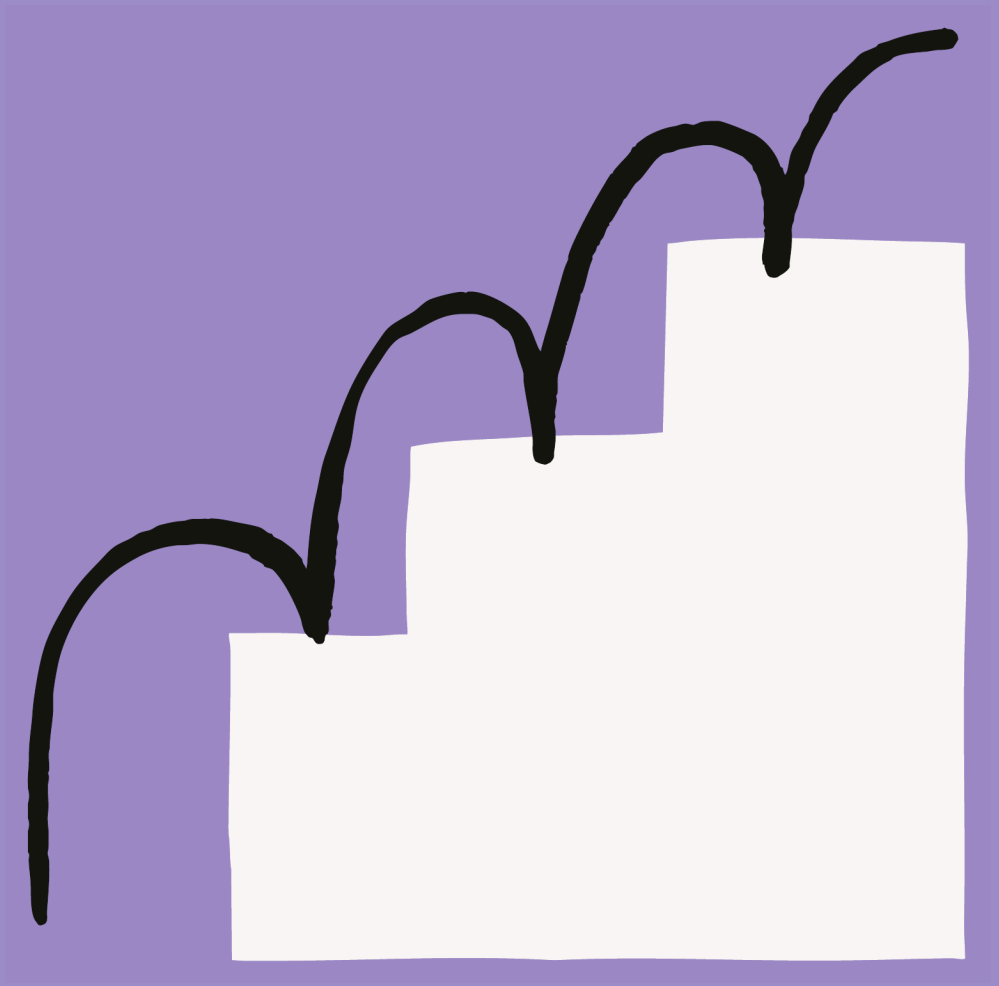
```
## Các vấn đề thường gặp
### Kết nối MCP thất bại
Nếu thấy "Connection refused":
1. Kiểm tra MCP server có đang chạy:
   vào Settings > Extensions
2. Xác nhận API key còn hợp lệ
3. Thử kết nối lại: Settings > Extensions
   > [Dịch vụ của bạn] > Reconnect
```

Trỏ rõ tới các tài nguyên đi kèm

```
Trước khi viết truy vấn, hãy xem
`references/api-patterns.md` để biết:
- Hướng dẫn về giới hạn tần suất gọi
- Cách phân trang
- Mã lỗi và cách xử lý
```

Tận dụng cơ chế tiết lộ theo từng lớp

Giữ SKILL.md tập trung vào hướng dẫn cốt lõi. Đưa tài liệu chi tiết sang `references/` và trỏ link tới đó. (Xem phần **Nguyên tắc thiết kế cốt lõi** để hiểu cơ chế ba lớp.)



Chương 3

Kiểm thử và cải tiến

Chương 3

Kiểm thử và cải tiến

Skill có thể được kiểm thử ở nhiều mức độ chặt chẽ khác nhau, tùy nhu cầu:

- **Kiểm thử thủ công trên Claude.ai** - chạy câu hỏi trực tiếp rồi quan sát. Lặp nhanh, không phải thiết lập gì.
- **Kiểm thử bằng script trên Claude Code** - tự động hóa các tình huống thử để kiểm lại được sau mỗi lần đổi.
- **Kiểm thử bằng lập trình qua skills API** - dựng bộ đánh giá chạy có hệ thống trên các tập kiểm thử đã định.

Hãy chọn cách hợp với yêu cầu chất lượng và mức độ phổ biến của skill. Một skill dùng nội bộ cho nhóm nhỏ có nhu cầu kiểm thử khác hẳn một skill triển khai cho hàng nghìn người dùng doanh nghiệp.

> **Mẹo hay: cải tiến trên một việc trước khi mở rộng**
> Chúng tôi nhận thấy người làm skill giỏi nhất thường cải tiến trên một việc khó duy nhất cho tới khi Claude làm được, rồi mới rút ra cách làm thắng cuộc đó thành skill. Cách này tận dụng khả năng học ngay trong ngữ cảnh của Claude và cho tín hiệu nhanh hơn so với thử dàn trải. Khi đã có nền tảng chạy được, hãy mở rộng ra nhiều tình huống thử để phủ rộng hơn.

Cách kiểm thử được khuyến nghị

Từ kinh nghiệm ban đầu, kiểm thử skill hiệu quả thường gồm ba mảng:

1. Kiểm thử việc kích hoạt

Mục tiêu: đảm bảo skill nạp đúng lúc.

Các trường hợp thử:

- ✓ Kích hoạt với những việc rõ ràng
- ✓ Kích hoạt cả khi yêu cầu được nói theo cách khác
- ✗ Không kích hoạt với chủ đề không liên quan

Ví dụ một bộ thử:

```
Nên kích hoạt:
- "Giúp tôi tạo workspace ProjectHub mới"
- "Tôi cần tạo một dự án trong ProjectHub"
- "Khởi tạo dự án ProjectHub cho kế hoạch quý 4"

KHÔNG nên kích hoạt:
- "Thời tiết ở San Francisco thế nào?"
- "Giúp tôi viết code Python"
- "Tạo một bảng tính" (trừ khi skill ProjectHub
  có lo cả bảng tính)
```

2. Kiểm thử chức năng

Mục tiêu: xác nhận skill cho ra kết quả đúng.

Các trường hợp thử:

- Tạo ra kết quả hợp lệ
- Các lần gọi API đều thành công
- Phần xử lý lỗi chạy đúng
- Bao quát các trường hợp biên

Ví dụ:

```
Kiểm thử: Tạo dự án với 5 nhiệm vụ
Đầu vào: tên dự án "Kế hoạch Q4", 5 mô tả nhiệm vụ
Khi: skill chạy quy trình
Kết quả mong đợi:
   - Dự án được tạo trong ProjectHub
   - 5 nhiệm vụ được tạo với thuộc tính đúng
   - Tất cả nhiệm vụ liên kết với dự án
   - Không có lỗi API
```

3. So sánh hiệu quả

Mục tiêu: chứng minh skill cho kết quả tốt hơn so với khi không có.

Dùng các chỉ số ở phần Xác định tiêu chí thành công. Một phép so sánh có thể trông như sau.

```
Khi không có skill:
- Người dùng phải đưa hướng dẫn mỗi lần
- 15 lượt trao đổi qua lại
- 3 lần gọi API thất bại, phải thử lại
- Tiêu tốn 12.000 token

Khi có skill:
- Quy trình chạy tự động
- Chỉ hỏi lại 2 câu cho rõ
- Không có lần gọi API nào thất bại
- Tiêu tốn 6.000 token
```

Dùng skill skill-creator

Skill skill-creator, có sẵn trên Claude.ai qua thư mục plugin hoặc tải về cho Claude Code, giúp bạn dựng và cải tiến skill. Nếu đã có MCP server và biết 2 đến 3 quy trình quan trọng nhất, bạn có thể dựng và kiểm thử một skill chạy được chỉ trong một lần ngồi làm, thường là 15 đến 30 phút.

Tạo skill:

- Sinh skill từ mô tả bằng lời tự nhiên
- Tạo SKILL.md đúng định dạng, kèm phần đầu YAML
- Gợi ý cụm từ kích hoạt và cấu trúc

Rà soát skill:

- Phát hiện lỗi thường gặp (description mơ hồ, thiếu điều kiện kích hoạt, cấu trúc có vấn đề)
- Chỉ ra nguy cơ kích hoạt quá mức hoặc thiếu
- Gợi ý tình huống thử dựa trên mục đích đã nêu của skill

Cải tiến dần:

- Sau khi dùng skill mà gặp trường hợp biên hay lỗi, hãy mang chính ví dụ đó quay lại skill-creator
- Ví dụ: "Dùng các vấn đề và giải pháp đã xác định trong cuộc trò chuyện này để cải thiện cách skill xử lý [trường hợp biên cụ thể]"

Cách dùng:

```
"Dùng skill skill-creator để giúp tôi xây một
skill cho [tình huống sử dụng của bạn]"
```

Lưu ý: skill-creator giúp bạn thiết kế và tinh chỉnh skill, nhưng không chạy các bộ kiểm thử tự động và không cho ra kết quả đánh giá định lượng.

Cải tiến dựa trên phản hồi

Skill là tài liệu sống. Hãy chuẩn bị tinh thần cải tiến dựa trên:

Dấu hiệu kích hoạt thiếu:

- Skill không nạp khi lẽ ra phải nạp
- Người dùng phải tự bật tay
- Có người hỏi hỗ trợ về việc khi nào nên dùng

Cách khắc phục: bổ sung chi tiết và sắc thái cho phần description, có thể thêm cả từ khóa, nhất là với các thuật ngữ kỹ thuật

Dấu hiệu kích hoạt quá mức:

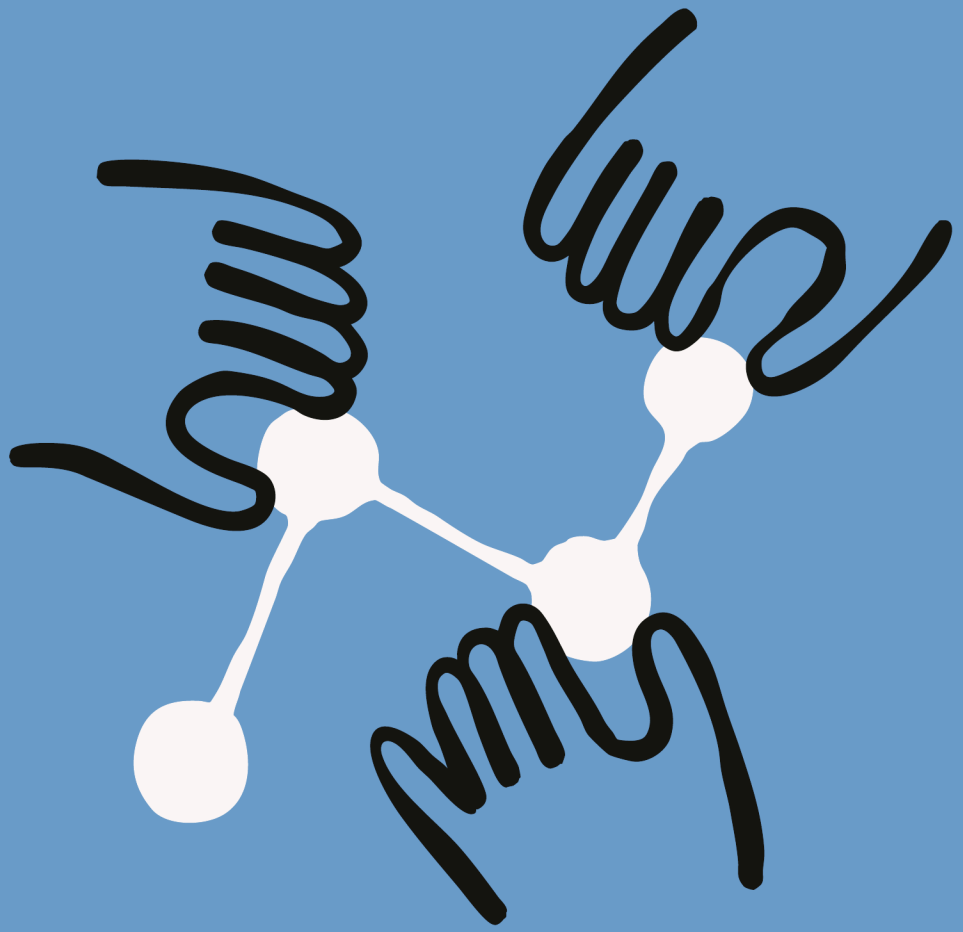
- Skill nạp cho cả những câu hỏi không liên quan
- Người dùng tắt nó đi
- Nhầm lẫn về mục đích của skill

Cách khắc phục: thêm điều kiện loại trừ, viết cụ thể hơn

Vấn đề khi chạy thực tế:

- Kết quả không nhất quán
- Lần gọi API thất bại
- Phải nhờ người dùng sửa

Cách khắc phục: viết hướng dẫn rõ hơn, thêm phần xử lý lỗi



Chương 4

Phân phối và chia sẻ

Chương 4

Phân phối và chia sẻ

Skill làm cho tích hợp MCP của bạn trọn vẹn hơn. Khi người dùng so sánh các connector, cái nào có skill sẽ đưa họ tới giá trị nhanh hơn, tạo lợi thế so với những lựa chọn chỉ có mỗi MCP.

Cách phân phối hiện nay (tháng 1/2026)

Người dùng cá nhân lấy skill thế nào:

- 1. Tải thư mục skill về
- 2. Nén thư mục (nếu cần)
- 3. Tải lên Claude.ai qua Settings > Capabilities > Skills
- 4. Hoặc đặt vào thư mục skills của Claude Code

Skill ở cấp tổ chức:

- Quản trị viên có thể triển khai skill cho cả workspace (ra mắt ngày 18/12/2025)
- Tự động cập nhật
- Quản lý tập trung

Một chuẩn mở

Chúng tôi đã công bố **Agent Skills** như một chuẩn mở. Giống MCP, chúng tôi tin skill nên dùng được trên nhiều công cụ và nền tảng: cùng một skill nên chạy được dù bạn dùng Claude hay nền tảng AI khác. Dù vậy, một số skill được thiết kế để khai thác tối đa năng lực của một nền tảng cụ thể; tác giả có thể ghi rõ điều đó trong trường compatibility. Chúng tôi đã cùng nhiều thành viên trong hệ sinh thái xây dựng chuẩn này, và rất phấn khởi trước mức độ đón nhận ban đầu.

Dùng skill qua API

Với các nhu cầu lập trình, như xây ứng dụng, agent, hay quy trình tự động có dùng skill, API cho bạn toàn quyền quản lý và chạy skill.

Năng lực chính:

- Endpoint `/v1/skills` để liệt kê và quản lý skill
- Thêm skill vào yêu cầu Messages API qua tham số `container.skills`
- Quản lý phiên bản qua Claude Console
- Dùng được với Claude Agent SDK để xây agent riêng

Khi nào dùng skill qua API thay vì Claude.ai:

Tình huống	Nơi phù hợp nhất
Người dùng cuối tương tác trực tiếp với skill	Claude.ai / Claude Code
Kiểm thử và cải tiến thủ công khi đang làm	Claude.ai / Claude Code
Việc lẻ, làm tới đâu hay tới đó	Claude.ai / Claude Code
Ứng dụng gọi skill bằng lập trình	API
Triển khai production ở quy mô lớn	API
Pipeline tự động và hệ thống agent	API

Lưu ý: skill trong API cần bản beta của Code Execution Tool, vốn cung cấp môi trường an toàn để skill chạy.

Chi tiết cách triển khai, xem:

- Skills API Quickstart
- Create Custom skills
- Skills in the Agent SDK

Cách làm được khuyến nghị hiện nay

Hãy bắt đầu bằng việc đưa skill lên GitHub với repo công khai, một README rõ ràng (cho người đọc, tách biệt với thư mục skill vốn không nên chứa README.md), kèm ví dụ sử dụng và ảnh chụp màn hình. Sau đó thêm vào tài liệu MCP một mục trỏ tới skill, giải thích vì sao dùng cả hai cùng nhau lại đáng, và kèm hướng dẫn bắt đầu nhanh.

- **1. Đưa lên GitHub**
  - Repo công khai cho skill mã nguồn mở
  - README rõ ràng kèm hướng dẫn cài đặt
  - Ví dụ sử dụng và ảnh chụp màn hình
- **2. Ghi tài liệu trong repo MCP của bạn**
  - Trỏ link tới skill từ tài liệu MCP
  - Giải thích giá trị khi dùng cả hai cùng nhau
  - Kèm hướng dẫn bắt đầu nhanh
- **3. Tạo hướng dẫn cài đặt**

```
## Cài đặt skill [Dịch vụ của bạn]

1. Tải skill về:
    - Clone repo: `git clone
      https://github.com/yourcompany/skills`
    - Hoặc tải file ZIP từ mục Releases

2. Cài vào Claude:
    - Mở Claude.ai > Settings > skills
    - Bấm "Upload skill"
    - Chọn thư mục skill (đã nén)

3. Bật skill:
    - Bật công tắc skill [Dịch vụ của bạn]
    - Đảm bảo MCP server đã kết nối

4. Kiểm thử:
    - Hỏi Claude: "Thiết lập một dự án mới
      trong [Dịch vụ của bạn]"
```

Định vị skill của bạn

Cách bạn mô tả skill quyết định việc người dùng có hiểu giá trị của nó và có chịu thử hay không. Khi viết về skill, dù trong README, tài liệu hay nội dung marketing, hãy nhớ mấy nguyên tắc sau.

Nói về kết quả, đừng nói về tính năng:

✓ **Tốt:**

```
"Skill ProjectHub giúp các nhóm dựng trọn
workspace dự án chỉ trong vài giây, gồm trang,
cơ sở dữ liệu và mẫu, thay vì mất 30 phút thiết
lập thủ công."
```

✗ **Tệ:**

```
"Skill ProjectHub là một thư mục chứa phần đầu
YAML và hướng dẫn Markdown để gọi các công cụ
MCP server của chúng tôi."
```

Làm nổi bật câu chuyện MCP cộng skill:

```
"MCP server của chúng tôi cho Claude truy cập
các dự án Linear của bạn. Skill của chúng tôi
dạy Claude quy trình lập kế hoạch sprint của
nhóm bạn. Kết hợp lại, chúng tạo nên năng lực
quản lý dự án bằng AI."
```

Chương 5

Các mẫu thiết kế và xử lý sự cố

Chương 5

Các mẫu thiết kế và xử lý sự cố

Những mẫu này đúc kết từ các skill do người dùng tiên phong và các nhóm nội bộ tạo ra. Chúng là những cách tiếp cận phổ biến mà chúng tôi thấy chạy tốt, không phải khuôn mẫu bắt buộc.

Chọn hướng tiếp cận: xuất phát từ vấn đề hay từ công cụ

Hãy hình dung như đi siêu thị đồ kim khí. Bạn có thể bước vào với một vấn đề ("tôi cần sửa cái tủ bếp") và nhân viên chỉ bạn tới đúng dụng cụ. Hoặc bạn chọn sẵn một cái khoan mới rồi hỏi cách dùng nó cho việc của mình.

Skill cũng vậy:

- **Xuất phát từ vấn đề:** "Tôi cần lập một workspace dự án" → Skill của bạn điều phối đúng các lệnh gọi MCP theo đúng trình tự. Người dùng mô tả kết quả mong muốn; skill lo phần công cụ.
- **Xuất phát từ công cụ:** "Tôi đã kết nối Notion MCP" → Skill của bạn dạy Claude các quy trình và cách làm tốt nhất. Người dùng có sẵn quyền truy cập; skill bổ sung chuyên môn.

Đa số skill nghiêng về một hướng. Biết hướng nào hợp với tình huống của mình sẽ giúp bạn chọn đúng mẫu bên dưới.

Mẫu 1: Điều phối quy trình tuần tự

Dùng khi: người dùng cần một quy trình nhiều bước theo đúng thứ tự.

Ví dụ cấu trúc:

```
## Quy trình: Tiếp nhận khách hàng mới

### Bước 1: Tạo tài khoản
Gọi công cụ MCP: `create_customer`
Tham số: name, email, company

### Bước 2: Thiết lập thanh toán
Gọi công cụ MCP: `setup_payment_method`
Chờ: xác minh phương thức thanh toán

### Bước 3: Tạo gói đăng ký
Gọi công cụ MCP: `create_subscription`
Tham số: plan_id, customer_id (từ Bước 1)

### Bước 4: Gửi email chào mừng
Gọi công cụ MCP: `send_email`
Mẫu: welcome_email_template
```

Kỹ thuật chính:

- Quy định rõ thứ tự các bước
- Các bước phụ thuộc lẫn nhau
- Kiểm tra ở từng giai đoạn
- Có hướng dẫn hoàn tác khi gặp lỗi

Mẫu 2: Phối hợp nhiều MCP

Dùng khi: quy trình trải qua nhiều dịch vụ.

Ví dụ: bàn giao từ thiết kế sang phát triển

```
### Giai đoạn 1: Xuất thiết kế (Figma MCP)
1. Xuất tài nguyên thiết kế từ Figma
2. Tạo bản đặc tả thiết kế
3. Tạo bản kê tài nguyên

### Giai đoạn 2: Lưu tài nguyên (Drive MCP)
1. Tạo thư mục dự án trong Drive
2. Tải lên toàn bộ tài nguyên
3. Tạo link chia sẻ

### Giai đoạn 3: Tạo nhiệm vụ (Linear MCP)
1. Tạo các nhiệm vụ phát triển
2. Đính link tài nguyên vào nhiệm vụ
3. Giao cho nhóm kỹ thuật

### Giai đoạn 4: Thông báo (Slack MCP)
1. Đăng tóm tắt bàn giao vào #engineering
2. Kèm link tài nguyên và tham chiếu nhiệm vụ
```

Kỹ thuật chính:

- Phân tách giai đoạn rõ ràng
- Truyền dữ liệu giữa các MCP
- Kiểm tra trước khi sang giai đoạn sau
- Xử lý lỗi tập trung

Mẫu 3: Tinh chỉnh theo vòng lặp

Dùng khi: chất lượng kết quả tốt dần qua mỗi vòng lặp.

Ví dụ: tạo báo cáo

```
## Tạo báo cáo theo vòng lặp

### Bản nháp đầu
1. Lấy dữ liệu qua MCP
2. Tạo bản nháp báo cáo đầu tiên
3. Lưu vào file tạm

### Kiểm tra chất lượng
1. Chạy script kiểm tra: `scripts/check_report.py`
2. Tìm các vấn đề:
    - Thiếu mục
    - Định dạng không nhất quán
    - Lỗi kiểm tra dữ liệu

### Vòng tinh chỉnh
1. Xử lý từng vấn đề đã tìm ra
2. Tạo lại các mục bị ảnh hưởng
3. Kiểm tra lại
4. Lặp đến khi đạt ngưỡng chất lượng

### Hoàn thiện
1. Áp định dạng cuối cùng
2. Tạo phần tóm tắt
3. Lưu bản cuối
```

Kỹ thuật chính:

- Tiêu chí chất lượng rõ ràng
- Cải thiện theo từng vòng
- Có script kiểm tra
- Biết khi nào nên dừng lặp

Mẫu 4: Chọn công cụ theo ngữ cảnh

Dùng khi: cùng một kết quả nhưng dùng công cụ khác nhau tùy ngữ cảnh.

Ví dụ: lưu trữ file

```
## Lưu file thông minh

### Cây quyết định
1. Kiểm tra loại và kích thước file
2. Chọn nơi lưu phù hợp nhất:
    - File lớn (>10MB): dùng MCP lưu trữ đám mây
    - Tài liệu cộng tác: dùng MCP Notion/Docs
    - File code: dùng MCP GitHub
    - File tạm: dùng lưu trữ cục bộ

### Thực hiện lưu
Tùy theo quyết định:
- Gọi công cụ MCP phù hợp
- Gắn metadata riêng theo dịch vụ
- Tạo link truy cập

### Giải thích cho người dùng
Nói rõ vì sao chọn nơi lưu đó
```

Kỹ thuật chính:

- Tiêu chí quyết định rõ ràng
- Có phương án dự phòng
- Minh bạch về lựa chọn

Mẫu 5: Hiểu biết chuyên ngành

Dùng khi: skill của bạn đóng góp kiến thức chuyên sâu vượt ra ngoài việc chỉ truy cập công cụ.

Ví dụ: tuân thủ trong tài chính

```
## Xử lý thanh toán có kiểm tra tuân thủ

### Trước khi xử lý (kiểm tra tuân thủ)
1. Lấy chi tiết giao dịch qua MCP
2. Áp các quy tắc tuân thủ:
    - Đối chiếu danh sách trừng phạt
    - Kiểm tra cho phép theo khu vực pháp lý
    - Đánh giá mức rủi ro
3. Ghi lại quyết định tuân thủ

### Xử lý
NẾU đạt tuân thủ:
    - Gọi công cụ MCP xử lý thanh toán
    - Áp các bước kiểm tra gian lận phù hợp
    - Xử lý giao dịch
NGƯỢC LẠI:
    - Đánh dấu để rà soát
    - Mở hồ sơ tuân thủ

### Nhật ký kiểm toán
- Ghi log mọi lần kiểm tra tuân thủ
- Lưu các quyết định xử lý
- Tạo báo cáo kiểm toán
```

Kỹ thuật chính:

- Chuyên môn ngành được cài vào trong logic
- Kiểm tra tuân thủ trước khi hành động
- Ghi chép đầy đủ
- Cơ chế quản trị rõ ràng

Xử lý sự cố

Không tải skill lên được

Lỗi: "Could not find SKILL.md in uploaded folder"

Nguyên nhân: file không được đặt tên đúng là SKILL.md

Cách khắc phục:

- Đổi tên thành `SKILL.md` (phân biệt hoa thường)
- Kiểm tra bằng `ls -la`, phải thấy SKILL.md

Lỗi: "Invalid frontmatter"

Nguyên nhân: sai định dạng YAML

Lỗi hay gặp:

```
# Sai - thiếu dấu phân cách
name: my-skill
description: Làm vài thứ

# Sai - thiếu dấu nháy đóng
name: my-skill
description: "Làm vài thứ

# Đúng
---
name: my-skill
description: Làm vài thứ
---
```

Lỗi: "Invalid skill name"

Nguyên nhân: tên có khoảng trắng hoặc chữ hoa

```
# Sai
name: My Cool Skill

# Đúng
name: my-cool-skill
```

Skill không kích hoạt

Biểu hiện: skill không bao giờ tự nạp

Cách sửa: chỉnh lại trường description. Xem phần Trường description để biết ví dụ tốt và tệ.

Kiểm nhanh:

- Có quá chung chung không? ("Giúp việc với các dự án" sẽ không ăn thua)
- Có chứa những câu mà người dùng thật sự sẽ nói không?
- Có nhắc loại file liên quan khi cần không?

Cách dò lỗi:

Hỏi Claude: "Bạn sẽ dùng skill [tên skill] trong trường hợp nào?" Claude sẽ đọc lại phần description. Dựa vào chỗ còn thiếu mà chỉnh.

Skill kích hoạt quá thường xuyên

Biểu hiện: skill nạp cho cả những câu hỏi không liên quan

Cách khắc phục:

1. Thêm điều kiện loại trừ

```
description: Phân tích dữ liệu nâng cao cho file
CSV. Dùng cho mô hình thống kê, hồi quy, phân
cụm. KHÔNG dùng cho việc khám phá dữ liệu đơn
giản (hãy dùng skill data-viz).
```

2. Viết cụ thể hơn

```
# Quá rộng
description: Xử lý tài liệu

# Cụ thể hơn
description: Xử lý tài liệu pháp lý PDF để rà
soát hợp đồng
```

3. Khoanh rõ phạm vi

```
description: Xử lý thanh toán PayFlow cho thương
mại điện tử. Chỉ dùng cho quy trình thanh toán
trực tuyến, không dùng cho các câu hỏi tài chính
chung.
```

Vấn đề kết nối MCP

Biểu hiện: skill nạp được nhưng lệnh gọi MCP thất bại

Kiểm theo danh sách:

- 1. Xác minh MCP server đã kết nối
  - Claude.ai: Settings > Extensions > [Dịch vụ của bạn]
  - Phải hiện trạng thái "Connected"
- 2. Kiểm tra xác thực
  - API key hợp lệ và chưa hết hạn
  - Đã cấp đúng quyền và phạm vi
  - Token OAuth đã được làm mới
- 3. Thử MCP riêng lẻ
  - Yêu cầu Claude gọi MCP trực tiếp (không qua skill)
  - "Dùng MCP [Dịch vụ] để lấy các dự án của tôi"
  - Nếu lỗi, vấn đề nằm ở MCP chứ không phải skill
- 4. Kiểm tra tên công cụ
  - Skill trỏ đúng tên công cụ MCP
  - Xem tài liệu của MCP server
  - Tên công cụ phân biệt hoa thường

Hướng dẫn không được làm theo

Biểu hiện: skill nạp được nhưng Claude không làm theo hướng dẫn

Nguyên nhân thường gặp:

- 1. Hướng dẫn quá dài dòng
  - Viết gọn lại
  - Dùng gạch đầu dòng và danh sách đánh số
  - Đưa phần tham khảo chi tiết ra file riêng
- 2. Hướng dẫn bị chìm
  - Đặt hướng dẫn quan trọng lên đầu
  - Dùng tiêu đề ## Important hoặc ## Critical
  - Lặp lại điểm then chốt nếu cần
- 3. Câu chữ mơ hồ

```
# Tệ
Nhớ kiểm tra mọi thứ cho cẩn thận

# Tốt
CRITICAL: Trước khi gọi create_project, kiểm tra:
- Tên dự án không được rỗng
- Có ít nhất một thành viên được giao
- Ngày bắt đầu không nằm trong quá khứ
```

Kỹ thuật nâng cao: với những bước kiểm tra quan trọng, hãy cân nhắc đóng gói một script tự kiểm tra bằng lập trình thay vì dựa vào hướng dẫn bằng lời. Code thì cho kết quả chắc chắn, còn diễn giải ngôn ngữ thì không. Xem các skill **Office** để có ví dụ về mẫu này.

4. Mô hình "làm cho qua" - thêm lời nhắc rõ ràng:

```
## Ghi chú về chất lượng
- Cứ làm thật kỹ, đừng vội
- Chất lượng quan trọng hơn tốc độ
- Đừng bỏ qua các bước kiểm tra
```

Lưu ý: thêm phần này vào prompt của người dùng sẽ hiệu quả hơn là để trong SKILL.md

Vấn đề khi ngữ cảnh quá lớn

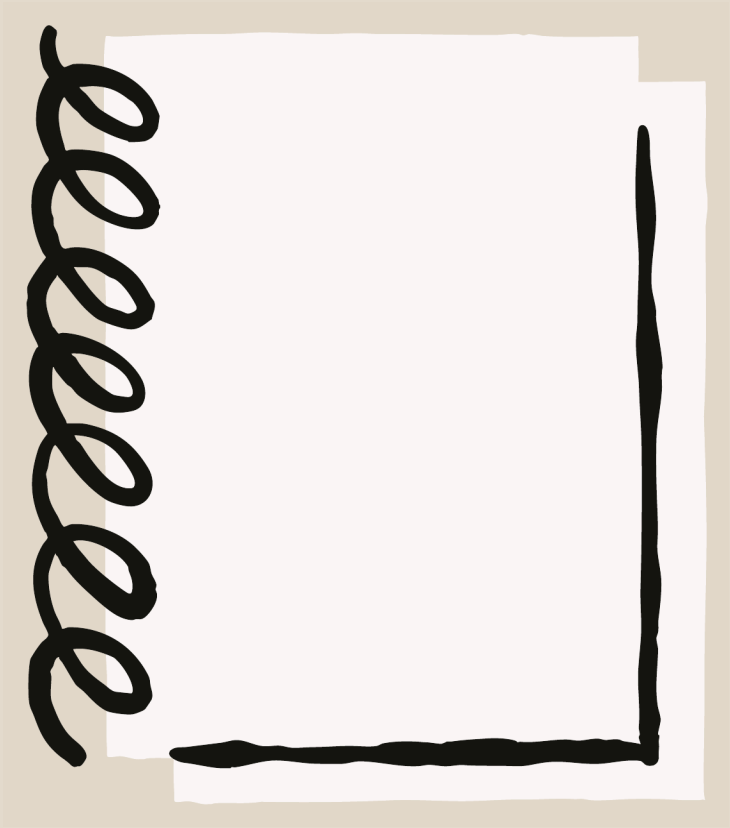
Biểu hiện: skill có vẻ chậm, hoặc câu trả lời kém đi

Nguyên nhân:

- Nội dung skill quá lớn
- Bật quá nhiều skill cùng lúc
- Nạp toàn bộ nội dung thay vì tiết lộ theo từng lớp

Cách khắc phục:

- 1. Tối ưu kích thước SKILL.md
  - Đưa tài liệu chi tiết sang references/
  - Trỏ link tới references thay vì viết thẳng vào file
  - Giữ SKILL.md dưới 5.000 từ
- 2. Giảm số skill đang bật
  - Xem lại xem có đang bật hơn 20 đến 50 skill cùng lúc không
  - Nên bật có chọn lọc
  - Cân nhắc gom skill liên quan thành từng "gói"



Chương 6

Tài nguyên và tham khảo

Tài nguyên và tham khảo

Nếu bạn đang làm skill đầu tiên, hãy bắt đầu với **Best Practices Guide**, rồi tra cứu tài liệu API khi cần.

Tài liệu chính thức

Tài nguyên của Anthropic:

- **Best Practices Guide**
- **Skills Documentation**
- **API Reference**
- **MCP Documentation**

Bài blog:

- **Introducing Agent Skills**
- **Engineering Blog: Equipping Agents for the Real World**
- **Skills Explained**
- **How to Create Skills for Claude**
- **Building Skills for Claude Code**
- **Improving Frontend Design through Skills**

Skill mẫu

Kho skill công khai:

- GitHub: **anthropics/skills**
- Chứa các skill do Anthropic tạo, bạn có thể tùy biến lại

Công cụ và tiện ích

Skill skill-creator:

- Tích hợp sẵn trong Claude.ai và có cho Claude Code
- Sinh được skill từ mô tả
- Rà soát và đưa ra khuyến nghị
- Cách dùng: "Giúp tôi xây một skill bằng skill-creator"

Kiểm tra:

- skill-creator có thể đánh giá skill của bạn
- Hỏi: "Rà soát skill này và gợi ý chỗ cải thiện"

Nhận hỗ trợ

Câu hỏi kỹ thuật:

- Câu hỏi chung: diễn đàn cộng đồng tại **Claude Developers Discord**

Báo lỗi:

- GitHub Issues: **anthropics/skills/issues**
- Kèm theo: tên skill, thông báo lỗi, các bước tái hiện

Phụ lục A: Danh sách kiểm tra nhanh

Dùng danh sách này để kiểm skill trước và sau khi tải lên. Muốn bắt đầu nhanh hơn, hãy dùng skill skill-creator để tạo bản nháp đầu, rồi rà qua danh sách này để chắc chắn không sót gì.

Trước khi bắt đầu

- ☐ Đã xác định 2-3 tình huống sử dụng cụ thể
- ☐ Đã xác định công cụ (có sẵn hoặc qua MCP)
- ☐ Đã đọc cẩm nang này và các skill mẫu
- ☐ Đã lên kế hoạch cấu trúc thư mục

Trong khi làm

- ☐ Thư mục đặt tên kebab-case
- ☐ Có file SKILL.md (đúng chính tả)
- ☐ Phần đầu YAML có dấu phân cách ---
- ☐ Trường name: kebab-case, không khoảng trắng, không hoa
- ☐ description có cả "làm gì" và "khi nào dùng"
- ☐ Không có thẻ XML (< >) ở bất kỳ đâu
- ☐ Hướng dẫn rõ ràng, làm được ngay
- ☐ Có phần xử lý lỗi
- ☐ Có ví dụ minh họa
- ☐ Các tài nguyên được trỏ link rõ ràng

Trước khi tải lên

- ☐ Đã thử kích hoạt với việc rõ ràng
- ☐ Đã thử kích hoạt với yêu cầu nói theo cách khác
- ☐ Đã xác minh không kích hoạt với chủ đề không liên quan
- ☐ Các bài kiểm thử chức năng đều đạt
- ☐ Tích hợp công cụ chạy được (nếu có)
- ☐ Đã nén thành file .zip

Sau khi tải lên

- ☐ Thử trong các cuộc trò chuyện thật
- ☐ Theo dõi kích hoạt thiếu hoặc quá mức
- ☐ Thu thập phản hồi người dùng
- ☐ Cải tiến phần description và hướng dẫn
- ☐ Cập nhật version trong metadata

Phụ lục B: Phần đầu YAML

Các trường bắt buộc

```
---
name: skill-name-in-kebab-case
description: Skill làm gì và khi nào dùng. Kèm
các cụm từ kích hoạt cụ thể.
---
```

Tất cả các trường tùy chọn

```
name: skill-name
description: [mô tả bắt buộc]
license: MIT # Tùy chọn: giấy phép cho mã nguồn mở
allowed-tools: "Bash(python:*) Bash(npm:*)
WebFetch" # Tùy chọn: giới hạn quyền dùng công cụ
metadata: # Tùy chọn: các trường tùy chỉnh
  author: Company Name
  version: 1.0.0
  mcp-server: server-name
  category: productivity
  tags: [project-management, automation]
  documentation: https://example.com/docs
  support: support@example.com
```

Ghi chú bảo mật

Được phép:

- Mọi kiểu dữ liệu YAML chuẩn (chuỗi, số, boolean, danh sách, đối tượng)
- Các trường metadata tùy chỉnh
- Mô tả dài (tối đa 1024 ký tự)

Bị cấm:

- Dấu ngoặc nhọn XML (< >) - giới hạn bảo mật
- Chạy code trong YAML (dùng cơ chế đọc YAML an toàn)
- Skill đặt tên bắt đầu bằng "claude" hoặc "anthropic" (đã giữ riêng)

Phụ lục C: Các ví dụ skill hoàn chỉnh

Để xem các skill đầy đủ, sẵn sàng cho production, minh họa những mẫu trong cẩm nang này:

- **Document Skills** - tạo **PDF**, **DOCX**, **PPTX**, **XLSX**
- **Example Skills** - nhiều mẫu quy trình khác nhau
- **Partner Skills Directory** - xem skill từ nhiều đối tác như Asana, Atlassian, Canva, Figma, Sentry, Zapier, và nhiều bên khác

Các kho này luôn được cập nhật và có thêm nhiều ví dụ ngoài những gì nêu ở đây. Hãy clone về, sửa lại cho tình huống của bạn, và dùng chúng làm mẫu.

Bản tiếng Việt được biên dịch và việt hóa bởi **Phong Hồ**.

✍️ **Theo dõi thêm bài viết & tài nguyên hữu ích về AI → phongminhho.substack.com**