

完整指南 构建技能指南 适用于 Claude



目录

引言 3 基础 4 规划与设计 7 测试与迭代 14 分发与共享 18 模式
与故障排除 21 资源与参考 28

引言

技能是一组指令——以简单文件夹的形式打包——用来教 Claude 如何处理特定任务或工作流。技能是为你的特定需求定制 Claude 的最强大方式之一。与其在每次对话中重复解释你的偏好、流程和领域专长，不如通过技能一次性教会 Claude，从此每次都能受益。

当你拥有可重复的工作流时，技能非常有用：根据规范生成前端设计、用一致的方法论进行研究、创建遵循团队风格指南的文档，或协调多步流程。它们能很好地与 Claude 的内置能力（例如代码执行和文档创建）配合使用。对于构建 MCP 集成的人来说，技能增加了另一个强大层级，有助于将原始的工具访问转化为可靠且优化的工作流。

本指南涵盖了构建高效技能所需了解的一切——从规划与结构到测试与分布。无论你是为自己、团队还是社区构建技能，都能在文中找到实用的模式和真实世界的示例。

你将学到：

- 技能结构的技术要求和最佳实践
- 独立技能和 MCP 增强工作流的模式
- 我们在不同用例中看到效果良好的模式
- 如何测试、迭代并分发您的技能

适合对象：

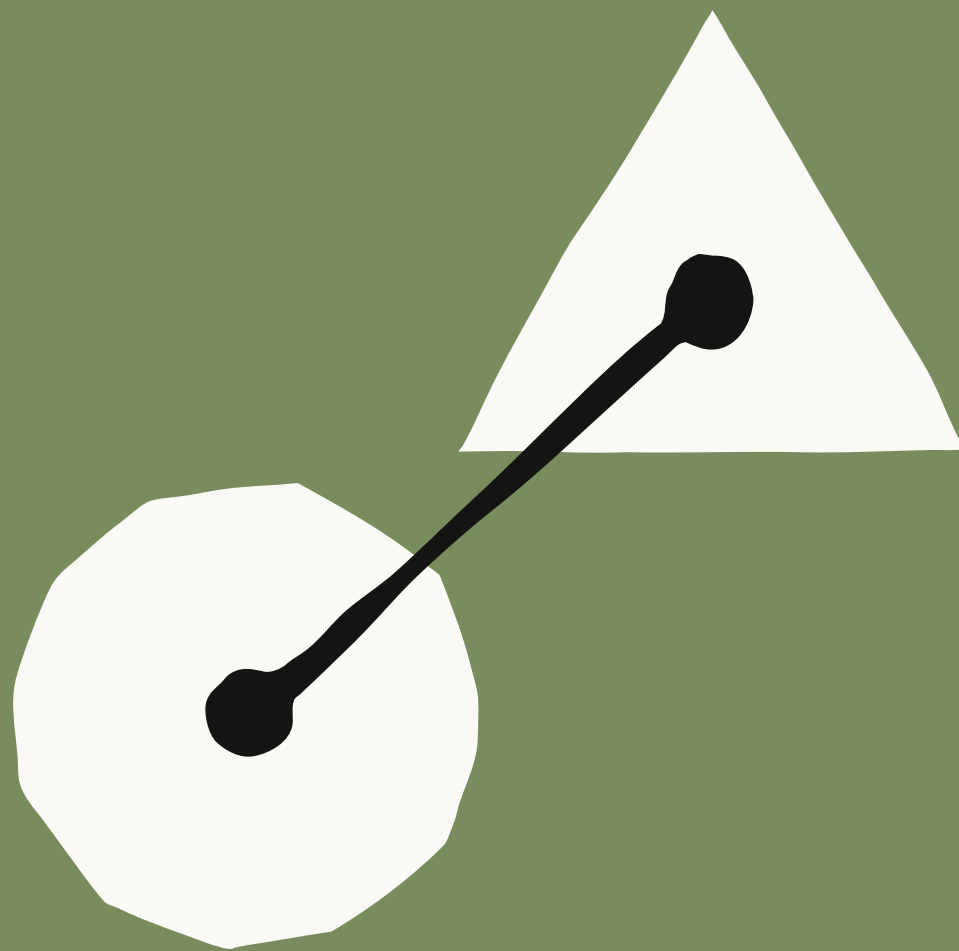
- 希望 Claude 在特定工作流中始终如一地执行的开发者
- 希望 Claude 遵循特定工作流的高级用户
- 希望在整个组织内标准化 Claude 工作方式的团队

通过本指南的两条路径

要构建独立技能吗？请侧重于基础、规划与设计，以及类别 1-2。要增强 MCP 集成？“Skills + MCP”部分和类别 3 适合你。两条路径共享相同的技术要求，但由你根据用例选择相关内容。

您将从本指南中获得：到最后，您将能够在一次会话中构建一个可用的 skill。使用 skill-creator 构建并测试第一个可运行 skill 大约需要 15-30 分钟。

我们开始吧。



第一章

基础

基础

什么是技能？

技能是一个包含以下内容的文件夹：

- SKILL.md（必需）：带有 YAML 文件头的 Markdown 指令
- scripts/（可选）：可执行代码（Python、Bash 等）
- references/ (optional): 按需加载的文档
- assets/ (optional): 输出中使用的模板、字体、图标

Core Design Principles

渐进披露

技能使用三级系统：

- 第一层级 (YAML 文件头)：始终加载到 Claude 的系统提示中。仅提供足够的信息以使 Claude 知晓何时应使用每项技能，而无需将全部内容加载到上下文中。
- Second level (SKILL.md body): 在 Claude 认为该技能与当前任务相关时加载。包含完整的指令和指导。
- Third level (Linked files): 附加文件，打包在技能目录内
这些文件只有在 Claude 需要时才会选择导航和发现。

这种渐进披露在维持专业专长的同时最大限度地减少了令牌使用量。

Composability

Claude 可以同时加载多个技能。你的技能应能与其他技能良好协作，不应假定自己是唯一可用的功能。

Portability

Skills 在 Claude.ai、Claude Code 和 API 上的工作方式相同。只需创建一次技能，它便可在所有平台上运行而无需修改，前提是运行环境支持该技能所需的任何依赖项。

For MCP 构建者：技能 + 连接器

💡 在没有 MCP 的情况下构建独立技能？跳到规划与设计 —— 你随时可以回到这里。

如果你已经拥有一个运行中的 MCP 服务器，那么艰难的部分已经完成。技能是位于其上的知识层——捕捉你已有的工作流和最佳实践，以便 Claude 能够一致地应用它们。

厨房类比

MCP 提供专业厨房：可使用的工具、原料和设备。

技能提供配方：逐步的指令，说明如何创建有价值的事物。

它们共同使用户能够完成复杂任务，而无需自己弄清楚每一个步骤。

它们如何协同工作：

MCP（连接性）	技能（知识）
将 Claude 连接到你的服务（Notion、Asana、Linear 等）	教会 Claude 如何有效使用你的服务
提供实时数据访问和工具调用	捕捉工作流和最佳实践
Claude 可以做什么	Claude 应该如何去做

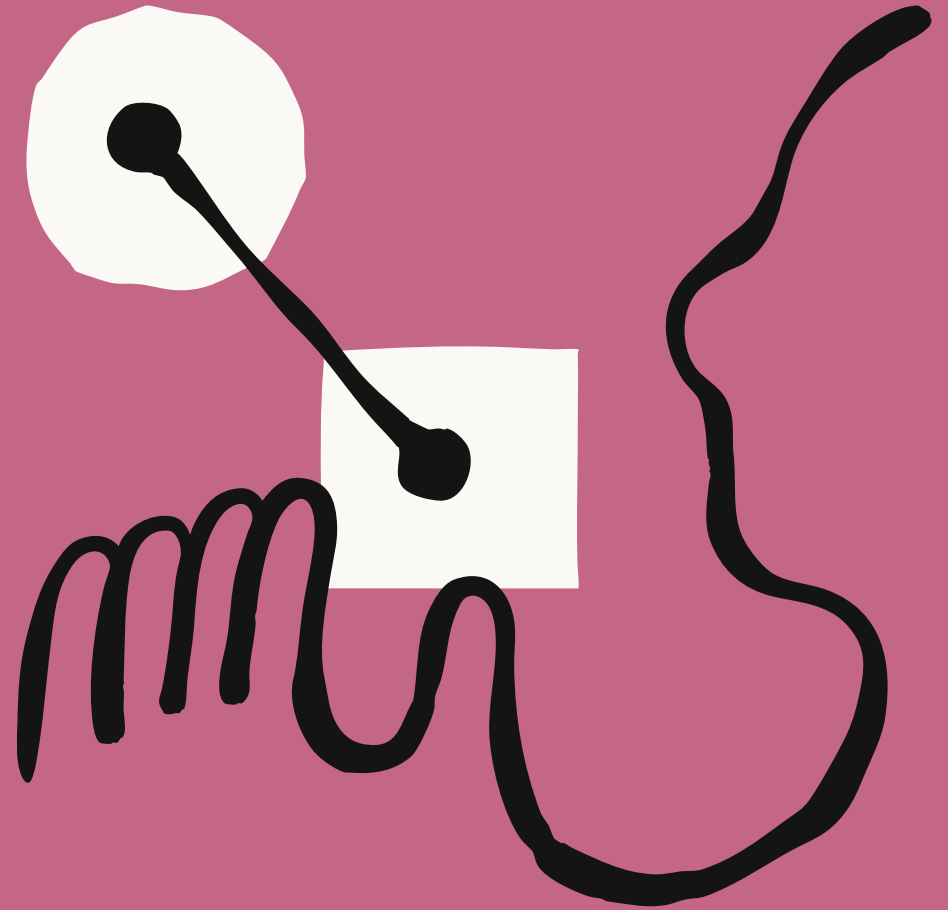
这对你的 MCP 用户为何重要

没有技能：

- 用户连接了你的 MCP，但不知道接下来该做什么
- 关于“我如何用你们的集成做 X”的支持工单
- 每次对话都从头开始
- 结果不一致，因为用户每次的提示不同
- 当真正的问题是工作流指导时，用户却指责你的连接器

有了技能：

- 预构建工作流在需要时自动激活
- 始终如一、可靠的工具使用
- 每次交互中都嵌入了最佳实践
- 降低您集成的学习曲线



第2章

规划与设计

规划与设计

从用例开始

在编写任何代码之前，先确定你的技能应实现的2-3个具体用例。

良好的用例定义：

```
Use Case: Project Sprint Planning
Trigger: User says "help me plan this sprint" or "create sprint tasks"
Steps:
1. Fetch current project status from Linear (via MCP)
2. Analyze team velocity and capacity
3. Suggest task prioritization
4. Create tasks in Linear with proper labels and estimates
Result: Fully planned sprint with tasks created
```

问问自己：

- 用户想要实现什么？
- 这需要哪些多步骤工作流？
- 需要哪些工具（内置还是 MCP? ）
- 应当嵌入哪些领域知识或最佳实践？

常见技能用例类别

在 Anthropic，我们观察到三种常见的用例：

类别 1：文档与资产创建

用途：用于创建一致且高质量的输出，包括文档、演示文稿、应用、设计、代码等。

真实示例：前端设计 技能（另见 docx、pptx、xlsx 和 ppt 的技能）

“创建具有高设计质量的独特、可投入生产的前端界面。用于构建网页组件、页面、构件、海报或应用程序时使用。”

关键技术：

- 嵌入式风格指南和品牌规范
- 用于一致输出的模板结构
- 在最终定稿前的质量检查清单
- 无需外部工具 - 使用 Claude 的内置能力

类别 2：工作流自动化

用于：受益于一致方法论的多步流程，包括跨多个 MCP 服务器的协调。

真实示例：skill-creator 技能

“创建新技能的交互式指南。引导用户完成用例定义、前言部分生成、指令编写和验证。”

关键技术：

- 逐步工作流与验证门控
- 常见结构的模板
- 内置审查和改进建议
- 迭代精化循环

类别 3：MCP 增强

用途：对 MCP 服务器提供的工具访问进行增强的工作流指导。

真实示例：sentry-code-review 技能（来自 Sentry）

"通过其 MCP 服务器，使用 Sentry 的错误监控数据自动分析并修复在 GitHub 拉取请求中检测到的错误。"

关键技术：

- 按顺序协调多个 MCP 调用
- 嵌入领域专长
- 提供用户否则需要指定的上下文
- 针对常见 MCP 问题的错误处理

定义成功标准

你如何判断你的技能在正常工作？

这些是愿景性目标——粗略的基准而非精确的门槛。力求严谨，但也要接受其中会包含一定基于直觉的评估成分。我们正在积极制定更健全的测量指南和工具。

定量指标：

- 技能在90%的相关查询中触发
 - 如何衡量：运行10–20个应该触发你技能的测试查询。跟踪它自动加载的次数与需要显式调用的次数。
- 在X次工具调用中完成工作流程
 - 如何衡量：比较在启用和未启用技能的情况下完成同一任务。统计工具调用次数和消耗的令牌总数。
- 0 每个工作流的失败 API 调用
 - 如何衡量：在测试运行期间监控 MCP 服务器日志。跟踪重试率和错误代码。

定性指标：

- 用户不需要提示 Claude 后续步骤
 - 如何评估：在测试中记录需要重定向或澄清的频率。向测试用户征求反馈。
- 工作流无需用户更正即可完成
 - 如何评估：对同一请求运行 3-5 次。比较输出的结构一致性和质量。
- 在不同会话中结果一致
 - 如何评估：新用户能否在首次尝试并在最少指导下完成任务？

技术要求

文件结构

```
your-skill-name/
├── SKILL.md           # Required - main skill file
├── scripts/          # Optional - executable code
│   ├── process_data.py # Example
│   └── validate.sh    # Example
├── references/        # Optional - documentation
│   ├── api-guide.md  # Example
│   └── examples/     # Example
└── assets/            # Optional - templates, etc.
    └── report-template.md # Example
```

关键规则

SKILL.md 命名:

- 必须完全为 **SKILL.md** (区分大小写)
- 不接受任何变体 (SKILL.MD、skill.md 等)

技能文件夹命名:

- 使用 kebab-case: **notion-project-setup** ✓
- 无空格: **Notion Project Setup** ✗
- 无下划线: **notion_project_setup** ✗
- 无大写字母: **NotionProjectSetup** ✗

不要有 README.md:

- 不要在你的技能文件夹内包含 README.md
- 所有文档都应放在 SKILL.md 或 references/
- 注意: 通过 GitHub 分发时, 你仍然需要为人工用户准备一个仓库级 README — 参见 分发与共享。

YAML 文件头: 最重要的部分

YAML 文件头决定 Claude 是否加载你的技能。请务必正确设置。

最小必需格式

```
---
name: your-skill-name
description: What it does. Use when user asks to [specific
phrases].
---
```

这就是开始所需的一切。

字段要求

name (必需):

- 仅限 kebab-case
- 不得有空格或大写字母
- 应与文件夹名称匹配

description (必需):

- 必须同时包含:
 - 技能的作用
 - 何时使用 (触发条件)
- 少于 1024 字符
- 无 XML 标签 (< 或 >)
- 包含用户可能说的具体任务
- 如相关, 提及文件类型

许可证（可选）：

- 如果要将 skill 开源，请使用
- 常见：MIT 许可证、Apache-2.0 许可证

兼容性（可选）

- 1-500 字符
- 指示环境要求：例如目标产品、所需系统包、网络访问需求等。

元数据（可选）：

- 任何自定义的键值对
- 建议：作者、版本、mcp-server

- 示例：

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

安全限制

前言部分禁止：

- XML 尖括号 (< >)
- 名称中包含"claude"或"anthropic"的技能（保留）

原因：前言部分出现在 Claude 的系统提示中。恶意内容可能注入指令。

编写有效的技能

描述字段

根据 Anthropic 工程博客："这些元数据……仅提供足够的信息以让 Claude 知道何时应使用每项技能，而无需将所有信息加载到上下文中。" 这是渐进披露的第一层级。

结构：

[What it does] + [When to use it] + [Key capabilities]

优秀描述示例：

```
# Good - specific and actionable
description: Analyzes Figma design files and generates
developer handoff documentation. Use when user uploads .figfiles,
asks for "design specs", "component documentation", or
"design-to-code handoff".

# Good - includes trigger phrases
description: Manages Linear project workflows including sprint
planning, task creation, and status tracking. Use when user
mentions "sprint", "Linear tasks", "project planning",
or asksto "create tickets".

# Good - clear value proposition
description: End-to-end customer onboarding workflow for
PayFlow. Handles account creation, payment setup, and
subscription management. Use when user says "onboard newcustomer",
"set up subscription", or "create PayFlow account".
```

示例：糟糕的描述：

```
# Too vague
description: Helps with projects.

# Missing triggers
description: Creates sophisticated multi-page documentation
systems.

# Too technical, no user triggers
description: Implements the Project entity model with
hierarchical relationships.
```

撰写主要指令

在前言部分之后，用 Markdown 撰写实际指令。

推荐结构：

针对您的技能调整此模板。将方括号部分替换为您的具体内容。

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

# Your Skill Name

## Instructions

### Step 1: [First Major Step]
Clear explanation of what happens.
```

```
Example:
```bash
python scripts/fetch_data.py --project-id PROJECT_ID
Expected output: [describe what success looks like]
```

（根据需要添加更多步骤）

## 示例

### 示例 1: [常见场景]

用户说：“创建一个新的营销活动”

操作：

1. 通过 MCP 获取现有的活动
2. 使用提供的参数创建新活动

结果：活动已创建并提供确认链接

（根据需要添加更多示例）

## 故障排除

错误：[Common error message]

原因：[Why it happens]

解决方案：[如何修复]

（根据需要添加更多错误情况）

## 指令的最佳实践

### 具体且可执行

✅ 良好示例:

```
Run `python scripts/validate.py --input {filename}` to check
data format.
If validation fails, common issues include:
- Missing required fields (add them to the CSV)
- Invalid date formats (use YYYY-MM-DD)
```

❌ 不良示例:

```
Validate the data before proceeding.
```

### 包含错误处理

```
Common Issues

MCP Connection Failed
If you see "Connection refused":
1. Verify MCP server is running: Check Settings > Extensions
2. Confirm API key is valid
3. Try reconnecting: Settings > Extensions > [Your Service] >
Reconnect
```

## 清晰引用打包资源

Before writing queries, consult `references/api-patterns.md` for:

- Rate limiting guidance
- Pagination patterns
- Error codes and handling

## 使用渐进披露

将 SKILL.md 保持为侧重核心指令的文件。将详细文档移至 `references/` 并链接到该处。（参见 [Core Design Principles](#) 以了解三层系统的工作方式。）



第3章

# 测试与迭代

# 测试与迭代

技能可以根据你的需求以不同严格程度进行测试：

- Manual testing in Claude.ai - 直接运行查询并观察行为。快速迭代，无需设置。
- Scripted testing in Claude Code - 自动化测试用例，以便在变更后进行可重复的验证。
- Programmatic testing via skills API - 构建评估套件，系统地针对定义的测试集运行。

选择与您的质量要求和技能可见性相匹配的方法。供小团队内部使用的技能与部署给数千名企业用户的技能在测试需求上有所不同。

专业提示：在扩展之前先对单个任务进行迭代

我们发现，最有效的技能创建者会围绕一个具有挑战性的任务反复迭代，直到 Claude 成功，然后将成功的方法提取为一个技能。这样可以利用 Claude 的上下文学习，并比广泛的测试提供更快反馈信号。一旦你有了可行的基础，就扩展到多个测试用例以覆盖更多情形。

## 推荐测试方法

根据早期经验，有效的技能测试通常涵盖三个方面：

### 1. 触发测试

目标：确保你的技能在正确的时间加载。

测试用例：

-  触发明显任务
-  触发于改写请求
-  不在无关主题上触发

示例测试套件：

Should trigger:

- "Help me set up a new ProjectHub workspace"
- "I need to create a project in ProjectHub"
- "Initialize a ProjectHub project for Q4 planning"

Should NOT trigger:

- "What's the weather in San Francisco?"
- "Help me write Python code"
- "Create a spreadsheet" (unless ProjectHub skill handles sheets)

## 2. 功能测试

目标：验证该技能产生正确的输出。

测试用例：

- 生成有效输出
- API 调用成功
- 错误处理正常
- 覆盖边缘情况

示例：

```
Test: Create project with 5 tasks
Given: Project name "Q4 Planning", 5 task descriptions
When: Skill executes workflow
Then:
 - Project created in ProjectHub
 - 5 tasks created with correct properties
 - All tasks linked to project
 - No API errors
```

## 3. 性能比较

目标：证明该技能相比基线能提升结果。

使用“定义成功标准”中的指标。下面是一个比较示例。

基线比较：

```
Without skill:
- User provides instructions each time
- 15 back-and-forth messages
- 3 failed API calls requiring retry
- 12,000 tokens consumed
```

```
With skill:
- Automatic workflow execution
- 2 clarifying questions only
- 0 failed API calls
- 6,000 tokens consumed
```

## 使用 skill-creator 技能

该 **skill-creator** 技能——可通过 Claude.ai 的插件目录获得，或下载用于 Claude Code——可以帮助您构建并迭代技能。如果您有 MCP 服务器并且知道您最常用的 2-3 个工作流，您可以在一次会话中构建并测试出一个可用的技能——通常只需 15-30 分钟。

创建技能：

- 从自然语言描述生成技能
- 生成带有前言部分的正确格式的 SKILL.md
- 建议触发短语和结构

审查技能：

- 标记常见问题（模糊描述、缺失触发、结构性问题）

- 识别潜在的过度/不足触发风险
- 根据技能的既定用途建议测试用例

迭代改进：

- 在使用你的 skill 时遇到边缘情况或失败后，请将这些示例反馈给 skill-creator
- 示例：“使用本次聊天中识别的问题与解决方案来改进该 skill 如何处理[specific edge case]”

使用方法：

```
"Use the skill-creator skill to help me build a skill for
[your use case]"
```

注意：skill-creator 帮助您设计和完善技能，但不会执行自动化测试套件或产生定量评估结果。

## 基于反馈的迭代

技能是活文档。请计划根据以下内容进行迭代：

触发不足信号：

- 技能在应加载时未加载
- 用户手动启用
- 关于何时使用的支持问题

解决方案：在描述中增加更多细节和细微差别——这可以包括尤其针对技术术语的关键词

触发过度信号：

- 技能针对不相关查询加载
- 用户禁用
- 对用途的混淆

解决方案：添加负触发器，变得更具体

执行问题：

- 结果不一致
- API 调用失败
- 需要用户更正

解决方案：改进指令，添加错误处理



#### 第4章

# 分发与共享

# 分发与共享

技能使您的 MCP 集成更加完整。当用户比较连接器时，具有技能的连接器能更快实现价值，为您相较于仅限MCP的替代方案带来优势。

## 当前分发模型（2026年1月）

个体用户如何获取技能：

- 1. 下载技能文件夹
- 2. 压缩文件夹（如有需要）
- 3. 通过 设置 > 功能 > 技能 将其上传到 Claude.ai
- 4. 或放置在 Claude Code 技能目录 中

组织级技能：

- 管理员 可以在整个工作区 部署 技能（于 2025年12月18日 发布）
- 自动更新
- 集中管理

## 一个开放标准

我们已发布 **Agent Skills** 作为一个开放标准。像 MCP 一样，我们相信技能应该可以在工具和平台之间移植——同一项技能应该能在无论你是使用 Claude 还是其他人工智能平台，都能工作。不过，有些技能是为充分利用特定平台的功能而设计的；作者可以在技能的 **compatibility** 字段中注明这点。我们一直在与生态系统的成员合作制定该标准，并对早期采纳感到振奋。

## 通过 API 使用技能

对于编程使用场景——例如构建应用程序、代理或利用技能的自动化工作流——API 提供了对技能管理和执行的直接控制。

主要功能：

- `/v1/skills` 用于列出和管理技能的端点
- 通过 `container.skills` 参数将技能添加到 Messages API 请求中
- 通过 Claude 控制台进行版本控制和管理
- 可与 Claude Agent SDK 配合使用以构建自定义代理

通过 API 使用技能 与 在 Claude.ai 使用技能 的何时：

用例	最佳呈现方式
终端用户直接与技能交互	Claude.ai / Claude Code
开发过程中的手动测试与迭代	Claude.ai / Claude Code
个体的临时工作流	Claude.ai / Claude Code
使用技能的应用程序（以编程方式）	API
大规模生产部署	API
自动化管道和代理系统	API

注意：API 中的技能需要 Code Execution Tool 测试版，该测试版为技能运行提供所需的安全环境。

有关实现细节，请参阅：

- [Skills API 快速入门](#)
- [创建自定义技能](#)
- [Agent SDK 中的技能](#)

## 当前推荐的方法

首先将你的技能托管在 GitHub 的公开仓库中，并提供清晰的 README（面向人类访客——这与技能文件夹分开，技能文件夹中不应包含 README.md），以及带有截图的示例用法。然后在你的 MCP 文档中添加一个章节，链接到该技能，说明将两者结合使用的价值，并提供一个快速入门指南。

### 1. 在 GitHub 托管

- 用于开源技能的公开仓库
- 清晰的自述文件，包含安装说明
- 示例用法和截图

### 2. 在你的 MCP 仓库中编写文档

- 从 MCP 文档链接到技能
- 解释将两者结合使用的价值
- 提供快速入门指南

### 3. 创建安装指南

```
Installing the [Your Service] skill

1. Download the skill:
 - Clone repo: `git clone https://github.com/yourcompany/skills`
 - Or download ZIP from Releases

2. Install in Claude:
 - Open Claude.ai > Settings > skills
 - Click "Upload skill"
```

- Select the skill folder (zipped)

#### 3. Enable the skill:

- Toggle on the [Your Service] skill
- Ensure your MCP server is connected

#### 4. Test:

- Ask Claude: "Set up a new project in [Your Service]"

## 为你的技能定位

你如何描述你的技能决定了用户是否理解其价值并实际尝试。在撰写有关你技能的内容时——无论是在你的自述文件、文档还是营销材料中——请记住以下原则。

关注结果，而非功能：

✅ 良好：

"The ProjectHub skill enables teams to set up complete project workspaces in seconds – including pages, databases, and templates – instead of spending 30 minutes on manual setup."

❌ 不良：

"The ProjectHub skill is a folder containing YAML frontmatter and Markdown instructions that calls our MCP server tools."

突出 MCP + 技能 故事：

"Our MCP server gives Claude access to your Linear projects. Our skills teach Claude your team's sprint planning workflow. Together, they enable AI-powered project management."



第5章

# 模式与故障排除

# 模式与故障排除

这些模式来自早期采用者和内部团队创建的技能。它们代表了 we 看到的常见且效果良好的方法，而非强制性的模板。

选择你的方法：以问题为先 vs. 以工具为先

可以把它想象成 Home Depot。你可能带着一个问题进来——“我需要修理一个厨房橱柜”——这时员工会指引你去合适的工具。或者你可能先挑了一个新电钻，然后询问如何用它完成你的具体工作。

技能的工作方式也是如此：

- 以问题为先：“我需要建立一个项目工作区” → 您的技能按正确顺序编排合适的 MCP 调用。用户描述结果；技能负责处理工具。
- 以工具为先：“我已将 Notion MCP 已连接” → 您的技能向 Claude 教授最优工作流和最佳实践。用户有访问权限；技能提供专业知识。

大多数技能偏向某一种方向。了解哪种框架适合您的用例有助于您在下面选择正确的模式。

## 模式 1：顺序 workflow 编排

使用场景：您的用户需要按特定顺序执行多步流程。

示例结构：

```
Workflow: Onboard New Customer

Step 1: Create Account
Call MCP tool: `create_customer`
Parameters: name, email, company

Step 2: Setup Payment
Call MCP tool: `setup_payment_method`
Wait for: payment method verification

Step 3: Create Subscription
Call MCP tool: `create_subscription`
Parameters: plan_id, customer_id (from Step 1)

Step 4: Send Welcome Email
Call MCP tool: `send_email`
Template: welcome_email_template
```

关键技术：

- 显式步骤顺序
- 步骤间依赖
- 每个阶段的验证
- 针对失败的回滚指令

## 模式 2：多 MCP 协调

适用场景： workflow 跨越多个服务。

示例：从设计到开发的移交

```
Phase 1: Design Export (Figma MCP)
1. Export design assets from Figma
2. Generate design specifications
3. Create asset manifest

Phase 2: Asset Storage (Drive MCP)
1. Create project folder in Drive
2. Upload all assets
3. Generate shareable links

Phase 3: Task Creation (Linear MCP)
1. Create development tasks
2. Attach asset links to tasks
3. Assign to engineering team

Phase 4: Notification (Slack MCP)
1. Post handoff summary to #engineering
2. Include asset links and task references
```

关键技术：

- 明确的阶段分离
- MCP 之间的数据传递
- 进入下一阶段前的验证
- 集中错误处理

## 模式 3：迭代改进

使用场景：当输出质量随着迭代而提高时。

示例：报告生成

```
Iterative Report Creation

Initial Draft
1. Fetch data via MCP
2. Generate first draft report
3. Save to temporary file

Quality Check
1. Run validation script: `scripts/check_report.py`
2. Identify issues:
 - Missing sections
 - Inconsistent formatting
 - Data validation errors

Refinement Loop
1. Address each identified issue
2. Regenerate affected sections
3. Re-validate
4. Repeat until quality threshold met

Finalization
1. Apply final formatting
2. Generate summary
3. Save final version
```

关键技术：

- 明确的质量标准
- 迭代改进
- 验证 脚本
- 知道何时停止迭代

## 模式4：上下文感知工具选择

使用时机：相同结果，但根据上下文选择不同的工具。

示例：文件存储

```
Smart File Storage

Decision Tree
1. Check file type and size
2. Determine best storage location:
 - Large files (>10MB): Use cloud storage MCP
 - Collaborative docs: Use Notion/Docs MCP
 - Code files: Use GitHub MCP
 - Temporary files: Use local storage

Execute Storage
Based on decision:
- Call appropriate MCP tool
- Apply service-specific metadata
- Generate access link

Provide Context to User
Explain why that storage was chosen
```

关键技术：

- 明确的决策标准
- 回退选项
- 关于选择的透明度

## 模式5：领域特定智能

WHEN（何时）使用：您的 skill 在提供工具访问之外，增加了专门知识。

示例：金融合规

```
Payment Processing with Compliance

Before Processing (Compliance Check)
1. Fetch transaction details via MCP
2. Apply compliance rules:
 - Check sanctions lists
 - Verify jurisdiction allowances
 - Assess risk level
3. Document compliance decision

Processing
IF compliance passed:
 - Call payment processing MCP tool
 - Apply appropriate fraud checks
 - Process transaction
ELSE:
 - Flag for review
 - Create compliance case

Audit Trail
- Log all compliance checks
- Record processing decisions
- Generate audit report
```

关键技术：

- 将领域专长嵌入逻辑中
- 先合规后行动
- 全面文档
- 明确治理

## 故障排除

### 技能无法上传

错误: "在上传的文件夹中找不到 SKILL.md"

原因: 文件未精确命名为 SKILL.md

解决方案:

- 重命名为 SKILL.md (区分大小写)
- 用以下命令验证: `ls -la` 应显示 SKILL.md

错误: "无效的文档头信息"

原因: YAML 格式问题

常见错误:

```
Wrong - missing delimiters
name: my-skill
description: Does things

Wrong - unclosed quotes
name: my-skill
description: "Does things

Correct

name: my-skill
description: Does things

```

错误: "无效的技能名称"

原因: 名称包含空格或大写字母

```
Wrong
name: My Cool Skill

Correct
name: my-cool-skill
```

### 技能未触发

症状: 技能从未自动加载

**Fix:**

修改你的描述字段。有关好的/不好的示例, 请参阅 描述字段。

Quick 检查清单:

- 是否过于笼统? (“有助于 projects” 不行)
- 是否包含用户实际会说的触发短语?
- 如果适用, 是否提到了相关的文件类型?

调试方法:

Ask Claude: "WHEN (何时) would you use the [技能名称] skill?" Claude will quote the 描述 back. Adjust based on what's missing.

### 技能触发率过高

症状: 技能在无关查询中加载

解决方案:

1. 添加负触发器

```
description: Advanced data analysis for CSV files. Use for
statistical modeling, regression, clustering. Do NOT use for
simple data exploration (use data-viz skill instead).
```

## 2. 更具体

```
Too broad
description: Processes documents

More specific
description: Processes PDF legal documents for contract review
```

## 3. 澄清范围

```
description: PayFlow payment processing for e-commerce. Use
specifically for online payment workflows, not for general
financial queries.
```

## MCP 连接问题

症状：技能加载成功但 MCP 调用失败

检查清单：

1. 验证 MCP 服务器已连接
  - Claude.ai：设置 > 扩展 > [您的服务]
  - 应显示“已连接”状态
2. 检查认证
  - API 密钥有效且未过期
  - 已授予适当的权限/作用域
  - OAuth 令牌已刷新
3. 单独测试 MCP
  - 让 Ask Claude 直接调用 MCP（不通过技能）
  - "使用 [Service] MCP 来获取我的 projects"
  - 如果失败，则问题出在 MCP 而非技能
4. 验证工具名称
  - 技能引用了正确的 MCP 工具名称
  - 检查 MCP 服务器文档
  - 工具名称区分大小写

## 未遵循指令

症状：技能已加载但 Claude 未遵循指令

常见原因：

1. 指令过于冗长
  - 保持指令简洁
  - 使用要点列举和编号列表
  - 将详细参考移到单独的文件中
2. 指令被淹没
  - 将关键指令放在顶部
  - 使用 ## Important 或 ## Critical 标题
  - 如有必要，重复关键点
3. 语言含糊不清

```
Bad
Make sure to validate things properly

Good
CRITICAL: Before calling create_project, verify:
- Project name is non-empty
- At least one team member assigned
- Start date is not in the past
```

高级技巧：对于关键验证，考虑捆绑一个以编程方式执行检查的脚本，而不是依赖语言指令。代码是确定性的；语言解释则不是。参见 Office 技能 中此模式的示例。

4. 模型“惰性” 添加明确的鼓励：

```
Performance Notes
- Take your time to do this thoroughly
- Quality is more important than speed
- Do not skip validation steps
```

注意：将此添加到用户提示中比添加到 SKILL.md 中更有效

## 大上下文问题

症状：技能似乎变慢或响应性能下降

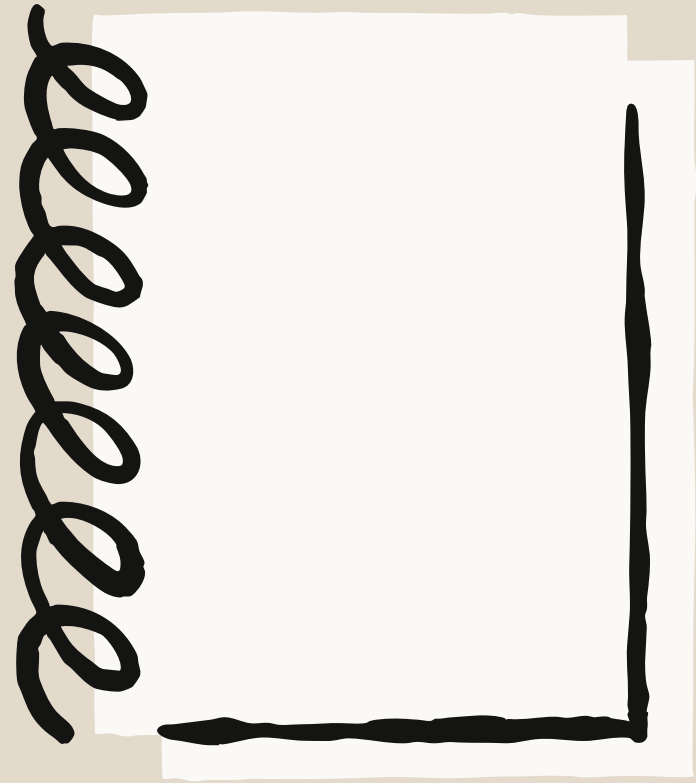
原因：

- 技能内容过大
- 同时启用的技能太多
- 加载了所有内容而不是渐进披露

解决方案：

1. 优化 SKILL.md 大小
  - 将详细文档移至 references/
  - 使用指向 references 的链接，而非内联
  - 保持 SKILL.md 少于 5,000 字
2. 减少已启用的技能
  - 评估是否同时启用了超过 20 - 50 个技能
  - 推荐选择性启用
  - 考虑为相关能力提供技能“包”

# 资源与参考



# 资源与参考

如果你要构建第一个技能，请先从最佳实践指南开始，然后在需要时参考 API 文档。

## 官方文档

### Anthropic 资源：

- [最佳实践指南](#)
- [技能文档](#)
- [API 参考](#)
- [MCP 文档](#)

### 博客文章：

- [介绍代理技能](#)
- [工程博客：为现实世界装备代理](#)
- [技能解析](#)
- [如何为 Claude 创建技能](#)
- [为 Claude Code 构建技能](#)
- [通过技能改进前端设计](#)

## 示例技能

### 公共技能代码仓库：

- GitHub: [anthropics/skills](#)
- 包含 Anthropic 创建的可定制技能

## 工具与实用程序

### skill-creator 技能：

- 内置于 Claude.ai 并可用于 Claude Code
- 可从描述生成技能
- 审查并提供建议
- 用法: "帮助我使用 skill-creator 构建一个 skill"

### 验证：

- skill-creator 可以评估你的技能
- 询问: "审查此 skill 并提出改进建议"

## 获取支持

### 有关技术问题：

- 常见问题: Claude 开发者 Discord 上的社区论坛

### 关于错误报告：

- GitHub Issues: [anthropics/skills/issues](#)
- 请包括: 技能名称、错误信息、重现步骤

# 参考 A: Quick checklist

在上传前后使用此检查清单来验证您的 skill。如果您想要更快地开始, 请使用 skill-creator 技能生成您的第一版草稿, 然后运行此清单以确保您没有遗漏任何内容。

## 开始之前

- ☐ 确定了 2-3 个具体用例
- ☐ 已识别工具 (内置或 MCP)
- ☐ 已查看本指南和示例技能
- ☐ 已规划文件夹结构

## 在开发期间

- ☐ 文件夹名称采用 kebab-case
- ☐ 存在 SKILL.md 文件 (精确拼写)
- ☐ YAML 文件头 使用 --- 分隔符
- ☐ name 字段: kebab-case, 不能有空格, 不能有大写字母
- ☐ 描述包含 WHAT (做什么) 和 WHEN (何时)
- ☐ 任何地方都不得出现 XML 标签 (<>)
- ☐ 指令清晰且可执行
- ☐ 错误处理 已包含
- ☐ 提供了示例
- ☐ references/ 已清晰链接

## 上传之前

- ☐ 在明显任务上测试触发
- ☐ 在改写请求上测试触发
- ☐ 验证未在无关主题上触发
- ☐ 功能测试通过
- ☐ 工具集成正常 (如适用)
- ☐ 压缩为 .zip 文件

## 上传后

- ☐ 在真实对话中测试
- ☐ 监测触发过少/过度触发
- ☐ 收集用户反馈
- ☐ 对描述和指令进行迭代
- ☐ 在元数据中更新版本

## 参考 B: YAML 文件头

### 必填字段

```

name: skill-name-in-kebab-case
description: What it does and when to use it. Include specific
trigger phrases.

```

### 所有可选字段

```
name: skill-name
description: [required description]
license: MIT # Optional: License for open-source
allowed-tools: "Bash(python:*) Bash(npm:*) WebFetch" # Optional:
Restrict tool access
metadata: # Optional: Custom fields
 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
```

### 安全说明

#### 允许:

- 任何标准 YAML 类型（字符串、数字、布尔值、列表、对象）
- 自定义元数据字段
- 长描述（最多 1024 字符）

#### 禁止:

- XML 尖括号 (<>) - 安全限制
- YAML 中的代码执行（使用安全的 YAML 解析）
- 以 “claude” 或 “anthropic” 前缀命名的技能（保留）

# 参考 C：完整技能示例

有关展示本指南模式的完整、可投入生产的技能：

- 文档技能 - [PDF](#), [DOCX](#), [PPTX](#), [XLSX](#) 创建
- [Example Skills](#) - Various workflow patterns
- [合作伙伴技能目录](#) - 查看来自 Asana 等不同合作伙伴的技能  
Atlassian、Canva、Figma、Sentry、Zapier 等更多

这些代码仓库会保持最新状态，并包含超出此处所述的其他示例。克隆它们，根据你的用例进行修改，并将它们用作模板。

