Claude 스킬 만들기

완벽 가이드

The Complete Guide to Building Skills for Claude

한국어 번역본

원문 다운로드 받기 클릭

목차

소개

스킬(Skill)이란 특정 작업이나 워크플로우를 Claude에게 처리하는 방법을 가르치기 위해, 간단한 폴더 형태로 패키징된 명령어 모음입니다. 스킬은 Claude를 여러분의 특정 요구에 맞게 커스터마이징하는 가장 강력한 방법 중 하나입니다. 대화할 때마다 여러분의 선호 방식, 작업 절차, 전문 지식을 반복해서 설명하는 대신, 스킬을 통해 한 번만 가르치고 매번 혜택을 누릴 수 있습니다.

스킬은 반복 가능한 워크플로우가 있을 때 특히 강력합니다. 예를 들어 사양서를 기반으로 프론트엔드 디자인 생성하기, 일관된 방법론으로 리서치 수행하기, 팀의 스타일 가이드를 따르는 문서 작성하기, 또는 다단계 프로세스 조율하기 등에 매우 유용합니다. 스킬은 코드 실행, 문서 생성 등 Claude의 내장 기능과도 잘 연동됩니다. MCP 통합을 구축하는 분들에게는 스킬이 또 하나의 강력한 레이어를 추가해주어, 단순한 도구 접근을 신뢰할 수 있는 최적화된 워크플로우로 발전시킵니다.

이 가이드는 계획 및 구조 설계부터 테스트 및 배포에 이르기까지 효과적인 스킬을 구축하는 데 필요한 모든 내용을 다룹니다. 개인용, 팀용, 또는 커뮤니티용 스킬을 구축하든 간에, 전반에 걸쳐 실용적인 패턴과 실제 예시를 확인할 수 있습니다.

학습 내용

- 스킬 구조에 관한 기술 요구사항 및 모범 사례
- 독립형 스킬과 MCP 연동 워크플로우 패턴
- 다양한 활용 사례에서 효과가 입증된 패턴들
- 스킬의 테스트, 반복 개선, 배포 방법

대상 독자

- Claude 가 특정 워크플로우를 일관되게 따르기를 원하는 개발자
- Claude 가 특정 워크플로우를 따르기를 원하는 파워 유저
- 조직 전반에 걸쳐 Claude 의 작동 방식을 표준화하고자 하는 팀

가이드 활용 방법

독립형 스킬을 구축하는 경우: 기본 개념, 계획 및 설계, 카테고리 1~2에 집중하세요. MCP 통합을 강화하

는 경우: '스킬 + MCP' 섹션과 카테고리 3이 핵심입니다. 두 경로 모두 동일한 기술 요구사항을 공유하지만, 각자의 상황에 맞는 부분을 선택하면 됩니다.

> 이 가이드를 끝까지 읽으면, 한 번에 앉아서 기능하는 스킬을 구축할 수 있게 됩니다. skill-creator를 활용하면 첫 번째 작동 스킬을 약 15~30분 내에 구축하고 테스트할 수 있습니다.

Chapter 1 기본 개념

스킬이란 무엇인가?

스킬은 다음 요소를 포함하는 폴더입니다:

- SKILL.md (필수): YAML 프론트매터가 있는 마크다운 형식의 명령어 파일
- scripts/ (선택): 실행 가능한 코드 (Python, Bash 등)
- references/ (선택): 필요에 따라 불러오는 참조 문서
- assets/ (선택): 출력물에 사용되는 템플릿, 폰트, 아이콘 등

핵심 설계 원칙

점진적 공개(Progressive Disclosure)

스킬은 3단계 시스템을 사용합니다:

- 1 단계 (YAML 프론트매터): Claude 의 시스템 프롬프트에 항상 로드됩니다. 모든 내용을 컨텍스트에 올리지 않고도 각 스킬을 언제 사용해야 하는지 Claude 가 파악할 수 있을 만큼의 최소 정보만 제공합니다.
- 2 단계 (SKILL.md 본문): 현재 작업에 스킬이 관련성이 있다고 판단될 때 로드됩니다. 전체 명령어와 가이드를 담고 있습니다.
- 3 단계 (연결된 파일들): 스킬 디렉토리 내에 번들로 포함된 추가 파일로, Claude 가 필요할 때만 선택적으로 탐색하고 확인합니다.

이 점진적 공개 방식은 토큰 사용량을 최소화하면서 전문화된 역량을 유지합니다.

조합 가능성(Composability)

Claude는 여러 스킬을 동시에 로드할 수 있습니다. 스킬은 유일한 기능이 아니라 다른 스킬과 함께 잘 작동하도록 설계해야 합니다.

이식성(Portability)

스킬은 Claude.ai, Claude Code, API 전반에 걸쳐 동일하게 작동합니다. 한 번 스킬을 만들면 환경이 해당 스킬의 의존성 요건을 지원하는 한, 수정 없이 모든 플랫폼에서 사용할 수 있습니다.

MCP 빌더를 위한 안내: 스킬 + 커넥터

> 💡 MCP 없이 독립형 스킬을 구축 중이라면 계획 및 설계 챕터로 바로 넘어가세요. 필요할 때 다시 돌아와도 됩니다.

이미 MCP 서버가 잘 작동하고 있다면 가장 어려운 부분은 끝난 것입니다. 스킬은 그 위에 쌓이는 지식 레이어로, 여러분이 이미 알고 있는 워크플로우와 모범 사례를 Claude가 일관되게 적용할 수 있도록 포착합니다.

주방 비유

MCP는 전문 주방을 제공합니다. 도구, 재료, 장비에 대한 접근권이죠.

스킬은 레시피를 제공합니다. 가치 있는 무언가를 만들기 위한 단계별 지침입니다.

두 가지가 합쳐지면, 사용자는 모든 단계를 직접 파악하지 않아도 복잡한 작업을 수행할 수 있게 됩니다.

MCP 와 스킬의 역할 비교

MCP (연결성)	스킬 (지식)
Claude를 서비스에 연결 (Notion, Asana, Linear 등)	서비스를 효과적으로 사용하는 방법을 Claude에게 가르침
실시간 데이터 접근 및 도구 실행 제공 Claude가 할 수 있는 것	워크플로우와 모범 사례 포착 Claude가 해야 하는 방법

스킬 없을 때 vs. 있을 때

스킬 없을 때:

- MCP 를 연결했지만 다음에 무엇을 해야 할지 모름
- "X 를 어떻게 하나요?"라는 지원 문의 발생
- 매 대화가 처음부터 시작

- 사용자마다 다르게 프롬프트를 입력해 결과가 일관성 없음

스킬 있을 때:

- 필요할 때 사전 구축된 워크플로우가 자동으로 활성화
- 일관되고 신뢰할 수 있는 도구 사용
- 모든 상호작용에 모범 사례 내재화
- 통합의 학습 곡선 단축

Chapter 2 계획 및 설계

활용 사례 정의부터 시작하기

코드를 작성하기 전에 스킬이 지원해야 하는 구체적인 활용 사례 2~3가지를 먼저 정의하세요.

좋은 활용 사례 정의 예시

```
활용 사례: 프로젝트 스프린트 계획
트리거: 사용자가 "이번 스프린트 계획 도와줘" 또는 "스프린트 작업 만들어줘"라고 말할 때
단계:
1. Linear에서 현재 프로젝트 상태 가져오기 (MCP를 통해)
2. 팀 속도 및 역량 분석
3. 작업 우선순위 제안
4. 적절한 라벨과 예상 시간이 포함된 작업을 Linear에 생성
결과: 모든 작업이 생성된 완전한 스프린트 계획
```

다음 질문들을 스스로에게 해보세요:

- 사용자가 달성하려는 것은 무엇인가?
- 이를 위해 어떤 다단계 워크플로우가 필요한가?
- 어떤 도구가 필요한가? (내장 기능 또는 MCP?)
- 어떤 도메인 지식이나 모범 사례를 포함해야 하는가?

일반적인 스킬 활용 사례 카테고리

Anthropic에서 가장 흔하게 관찰되는 세 가지 활용 사례입니다.

카테고리 1: 문서 및 에셋 생성

용도: 문서, 프레젠테이션, 앱, 디자인, 코드 등 일관성 있는 고품질 산출물 생성

실제 예시: frontend-design 스킬 (docx, pptx, xlsx, ppt 스킬도 참고)

```
"사양서로부터 독창적이고 프로덕션 수준의 프론트엔드 인터페이스를 높은 디자인 품질로 생성합니다.
웹 컴포넌트, 페이지, 아티팩트, 포스터 또는 애플리케이션 구축 시 사용하세요."
```

핵심 기법:

- 스타일 가이드 및 브랜드 기준 내장
- 일관된 산출물을 위한 템플릿 구조
- 완료 전 품질 체크리스트
- 외부 도구 불필요 — Claude 의 내장 기능만 사용

카테고리 2: 워크플로우 자동화

용도: 일관된 방법론이 필요한 다단계 프로세스, 여러 MCP 서버 간 조율 포함

실제 예시: skill-creator 스킬

> "새 스킬 생성을 위한 대화형 가이드. 활용 사례 정의, 프론트매터 생성, 명령어 작성, 검증 단계를 차례로 안내합니다."

핵심 기법:

- 검증 게이트가 있는 단계별 워크플로우
- 일반적인 구조를 위한 템플릿
- 내장된 검토 및 개선 제안
- 반복적 정제 루프

카테고리 3: MCP 강화

용도: MCP 서버가 제공하는 도구 접근을 강화하는 워크플로우 가이드

실제 예시: sentry-code-review 스킬 (Sentry 제공)

> "Sentry의 MCP 서버를 통해 에러 모니터링 데이터를 활용해 GitHub PR에서 감지된 버그를 자동으로 분석하고 수정합니다."

핵심 기법:

- 여러 MCP 호출을 순서대로 조율
- 도메인 전문 지식 내장
- 사용자가 별도로 지정해야 했을 맥락 자동 제공
- 일반적인 MCP 오류 처리

성공 기준 정의

스킬이 제대로 작동하는지 어떻게 알 수 있을까요?

이 지표들은 정밀한 임계값이 아닌 방향성을 제시하는 기준입니다. 엄밀함을 추구하되, 어느 정도의 주관적 판단도 허용하세요. Anthropic은 더 강력한 측정 가이드와 도구를 지속적으로 개발하고 있습니다.

정량적 지표

- 관련 쿼리의 90%에서 스킬 트리거: 스킬을 트리거해야 하는 테스트 쿼리 10~20 개를 실행하여 자동 로드되는 횟수를 추적하세요.
- 워크플로우를 X 번의 도구 호출 내에 완료: 스킬 유무에 따라 동일 작업을 비교하고 도구 호출 횟수와 토큰 소비량을 측정하세요.
- 워크플로우당 API 호출 실패 0 건: 테스트 중 MCP 서버 로그를 모니터링하고 재시도율 및 오류 코드를 추적하세요.

정성적 지표

- 사용자가 다음 단계를 직접 프롬프트할 필요 없음: 테스트 중 방향 재설정이나 설명이 필요한 빈도를 기록하고 베타 사용자 피드백을 수집하세요.
- 사용자 수정 없이 워크플로우 완료: 동일한 요청을 3~5 회 실행하고 구조적 일관성과 품질을 비교하세요.
- 세션 간 일관된 결과: 새 사용자가 최소한의 안내로 처음 시도에 작업을 완수할 수 있는지 확인하세요.

기술적 요구사항

파일 구조

```
your-skill-name/
├── SKILL.md          # 필수 – 메인 스킬 파일
├── scripts/          # 선택 – 실행 가능한 코드
│   ├── process_data.py
│   └── validate.sh
├── references/       # 선택 – 문서
│   ├── api-guide.md
│   └── examples/
└── assets/           # 선택 – 템플릿 등
    └── report-template.md
```

필수 규칙

SKILL.md 파일명:

- 정확히 SKILL.md 여야 합니다 (대소문자 구분)
- SKILL.MD, skill.md 등 다른 형태는 허용되지 않습니다

스킬 폴더 이름:

- kebab-case 사용: notion-project-setup ☑
- 공백 없음: Notion Project Setup ❌
- 밑줄 없음: notion_project_setup ❌
- 대문자 없음: NotionProjectSetup ❌

README.md 없음:

- 스킬 폴더 내에 README.md 를 포함하지 마세요
- 모든 문서는 SKILL.md 또는 references/에 작성하세요
- GitHub 을 통해 배포할 경우, 인간 사용자를 위한 레포지토리 수준의 README 는 별도로 작성하세요

YAML 프론트매터: 가장 중요한 부분

YAML 프론트매터는 Claude가 스킬을 로드할지 결정하는 방법입니다. 이 부분을 제대로 작성하세요.

최소 필수 형식:

```
---
name: your-skill-name
description: 이 스킬이 무엇을 합니다. 사용자가 [특정 문구]를 말할 때 사용하세요.
---
```

이것만으로 시작할 수 있습니다.

필드 요구사항

name (필수):

- kebab-case 만 사용
- 공백이나 대문자 없음
- 폴더 이름과 일치해야 함

description (필수):

- 반드시 '무엇을 하는지'와 '언제 사용하는지(트리거 조건)' 두 가지를 포함해야 함
- 1024 자 이하
- XML 태그 (< 또는 >) 없음
- 사용자가 실제로 말할 수 있는 구체적인 작업 포함
- 관련이 있다면 파일 유형 언급

license (선택): 오픈소스로 공개할 때 사용 (예: MIT, Apache-2.0)

compatibility (선택): 환경 요구사항 표시 (1~500자)

metadata (선택): 커스텀 키-값 쌍 (author, version, mcp-server 등)

보안 제한사항

프론트매터에서 금지된 항목:

- XML 꺾쇠 괄호 (< >)
- 이름에 'claude' 또는 'anthropic' 포함 (예약어)

이유: 프론트매터는 Claude의 시스템 프롬프트에 나타납니다. 악의적인 콘텐츠는 명령어를 주입할 수 있습니다.

효과적인 스킬 작성

description 필드

Anthropic 엔지니어링 블로그에 따르면: 이 메타데이터는 "모든 내용을 컨텍스트에 로드하지 않고도 Claude가 각 스킬을 언제 사용해야 하는지 파악할 수 있을 만큼의 정보만 제공"합니다. 이것이 점진적 공개의 첫 번째 단계입니다.

구조: [무엇을 하는지] + [언제 사용하는지] + [핵심 기능]

좋은 description 예시:

```
# 좋음 – 구체적이고 실행 가능
description: Figma 디자인 파일을 분석하고 개발자 핸드오프 문서를 생성합니다.
사용자가 .fig 파일을 업로드하거나, "디자인 스펙", "컴포넌트 문서",
"디자인-코드 핸드오프"를 요청할 때 사용하세요.
```

```
# 좋음 – 트리거 문구 포함
description: 스프린트 계획, 작업 생성, 상태 추적 등 Linear 프로젝트 워크플로우를
관리합니다. 사용자가 "스프린트", "Linear 작업", "프로젝트 계획"을 언급하거나
"티켓 만들어줘"라고 할 때 사용하세요.
```

나쁜 description 예시:

```
# 나쁨 – 너무 모호함
description: 프로젝트를 도와줍니다.
```

```
# 나쁨 – 트리거 조건 없음
description: 정교한 다중 페이지 문서 시스템을 생성합니다.
```

주요 명령어 작성하기

프론트매터 다음에 마크다운 형식으로 실제 명령어를 작성하세요. 다음 구조를 권장합니다:

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


# 스킬 이름
```

```
## 1단계: [첫 번째 주요 단계]
단계에 대한 명확한 설명


## 예시
사용자 말: "새 마케팅 캠페인 설정해줘"
실행 액션: 1. MCP로 기존 캠페인 가져오기 2. 새 캠페인 생성


## 트러블슈팅
오류: [일반적인 오류 메시지]
원인: [왜 발생하는지]
해결: [어떻게 수정하는지]
```

명령어 작성 모범 사례

구체적이고 실행 가능하게 작성하기:

```
☑ 좋음:
데이터 형식을 확인하려면 `python scripts/validate.py --input {filename}`을 실행하세요.
검증 실패 시 일반적인 문제:
- 필수 필드 누락 (CSV에 추가하세요)
- 잘못된 날짜 형식 (YYYY-MM-DD 형식을 사용하세요)


❌ 나쁨:
계속 진행하기 전에 데이터를 검증하세요.
```

에러 처리 포함:

```
## 자주 발생하는 오류
### MCP 연결 실패
"Connection refused"가 표시되면:
1. MCP 서버 실행 확인: 설정 > 확장 프로그램 확인
2. API 키가 유효한지 확인
3. 재연결 시도: 설정 > 확장 프로그램 > [서비스명] > 재연결
```

Chapter 3 테스트와 반복 개선

스킬은 필요에 따라 다양한 수준의 엄밀함으로 테스트할 수 있습니다:

- Claude.ai 에서 수동 테스트 — 쿼리를 직접 실행하고 동작을 관찰합니다. 빠른 반복, 별도 설정 불필요
- Claude Code 에서 스크립트 테스트 — 변경 사항에 걸쳐 반복 가능한 검증을 위해 테스트 케이스를 자동화합니다
- Skills API 를 통한 프로그래매틱 테스트 — 정의된 테스트 세트에 대해 체계적으로 실행되는 평가 모음을 구축합니다

스킬의 품질 요구사항과 노출 범위에 맞는 접근 방식을 선택하세요. 소규모 팀이 내부적으로 사용하는 스킬과 수천 명의 엔터프라이즈 사용자에게 배포되는 스킬은 다른 테스트 요구사항을 가집니다.

> 💡 Pro Tip: 하나의 작업을 완성한 후 확장하세요 가장 효과적인 스킬 제작자들은 Claude가 성공할 때까지 하나의 도전적인 작업을 반복 개선한 후, 성공한 접근 방식을 스킬로 추출합니다. 이는 Claude의 컨텍스트 내 학습을 활용하며, 광범위한 테스트보다 더 빠른 신호를 제공합니다.

권장 테스트 접근법

효과적인 스킬 테스트는 일반적으로 세 가지 영역을 다룹니다:

1. 트리거 테스트

목표: 스킬이 적절한 시점에 로드되는지 확인

테스트 케이스:

- ☑ 명확한 작업에서 트리거됨
- ☑ 다른 표현으로 바꾼 요청에서 트리거됨
- ✗ 관련 없는 주제에서는 트리거되지 않음

예시:

```
트리거되어야 하는 경우:
- "새 ProjectHub 워크스페이스 설정해줘"
```

```
- "ProjectHub에서 프로젝트 만들어야 해"
- "Q4 계획을 위한 ProjectHub 프로젝트 초기화해줘"


트리거되어서는 안 되는 경우:
- "샌프란시스코 날씨 어때?"
- "Python 코드 작성 도와줘"
- "스프레드시트 만들어줘" (ProjectHub 스킬이 다루지 않는 경우)
```

2. 기능 테스트

목표: 스킬이 올바른 결과를 생성하는지 검증

테스트 케이스:

- 유효한 결과물 생성
- API 호출 성공
- 에러 처리 작동
- 엣지 케이스 처리

예시:

```
테스트: 5개 작업이 있는 프로젝트 생성
조건: 프로젝트명 "Q4 Planning", 5개 작업 설명
실행 시:
결과:
  - ProjectHub에 프로젝트 생성됨
  - 올바른 속성으로 5개 작업 생성됨
  - 모든 작업이 프로젝트에 연결됨
  - API 오류 없음
```

3. 성능 비교

목표: 스킬이 기준 대비 결과를 개선함을 증명

비교 예시:

```
스킬 없을 때:
- 매번 사용자가 직접 명령어 제공
- 15번의 질문-답변 반복
- 재시도가 필요한 API 호출 실패 3건
- 토큰 12,000개 소비
```

```
스킬 있을 때:
- 자동 워크플로우 실행
- 명확화 질문 2개만 필요
- API 호출 실패 0건
- 토큰 6,000개 소비
```

skill-creator 스킬 활용하기

skill-creator 스킬 — Claude.ai 플러그인 디렉토리에서 사용 가능하거나 Claude Code용으로 다운로드 가능 — 은 스킬을 구축하고 반복 개선하는 데 도움을 줍니다. MCP 서버와 주요 워크플로우 2~3가지를 알고 있다면, 한 번에 앉아서 약 15~30분 내에 기능하는 스킬을 구축하고 테스트할 수 있습니다.

스킬 생성:

- 자연어 설명으로 스킬 생성
- 올바른 형식의 SKILL.md 와 프론트매터 생성
- 트리거 문구 및 구조 제안

스킬 검토:

- 일반적인 문제 표시 (모호한 설명, 누락된 트리거, 구조 문제 등)
- 과도한/부족한 트리거 위험 식별
- 스킬의 목적에 기반한 테스트 케이스 제안

반복적 개선:

- 스킬을 사용하다 엣지 케이스나 실패를 만나면, 해당 예시를 skill-creator 에 가져와 개선하세요
- 예시: "이 대화에서 발견된 문제와 해결책을 활용해 스킬이 [특정 엣지 케이스]를 처리하는 방식을 개선해줘"

```
사용 방법:
"skill-creator 스킬을 사용해 [활용 사례]를 위한 스킬 만드는 것을 도와줘"
```

참고: skill-creator는 스킬 설계와 정제를 도와주지만, 자동화된 테스트 스위트를 실행하거나 정량적 평가 결과를 생성하지는 않습니다.

피드백 기반 반복 개선

스킬은 살아있는 문서입니다. 다음을 바탕으로 반복 개선할 계획을 세우세요:

트리거 부족 신호:

- 스킬이 필요할 때 로드되지 않음
- 사용자가 수동으로 활성화
- 언제 사용하는지에 대한 지원 문의

해결책: description에 더 많은 세부 사항과 구체적인 키워드를 추가하세요

트리거 과잉 신호:

- 관련 없는 쿼리에서 스킬이 로드됨
- 사용자가 스킬을 비활성화함
- 목적에 대한 혼란

해결책: 부정적 트리거를 추가하거나 더 구체적으로 작성하세요

실행 문제:

- 일관성 없는 결과
- API 호출 실패
- 사용자 수정 필요

해결책: 명령어를 개선하고 에러 처리를 추가하세요

Chapter 4 배포 및 공유

스킬은 MCP 통합을 더욱 완성도 있게 만들어줍니다. 사용자들이 커넥터를 비교할 때, 스킬을 제공하는 커넥터가 MCP만 제공하는 대안보다 더 빠르게 가치를 제공합니다.

현재 배포 모델 (2026 년 기준)

개인 사용자가 스킬을 얻는 방법:

1. 스킬 폴더 다운로드
2. 폴더 압축 (필요한 경우)
3. Claude.ai 설정 > 기능 > 스킬에서 업로드
4. 또는 Claude Code 스킬 디렉토리에 배치

조직 수준 스킬:

- 관리자는 전사적으로 스킬을 배포 가능
- 자동 업데이트
- 중앙 집중식 관리

오픈 표준

Anthropic은 Agent Skills를 오픈 표준으로 공개했습니다. MCP와 마찬가지로, 스킬은 도구와 플랫폼 간에 이식 가능해야 한다고 생각합니다. 동일한 스킬이 Claude든 다른 AI 플랫폼이든 상관없이 작동해야 합니다. 단, 일부 스킬은 특정 플랫폼의 기능을 최대한 활용하도록 설계될 수도 있으며, 이 경우 compatibility 필드에 명시할 수 있습니다.

API 를 통한 스킬 사용

애플리케이션, 에이전트, 스킬을 활용하는 자동화된 워크플로우 구축 등 프로그래매틱 활용 사례에는 API가 직접적인 제어 방법을 제공합니다.

주요 기능:

- 스킬 목록 조회 및 관리를 위한 /v1/skills 엔드포인트
- container.skills 매개변수를 통해 Messages API 요청에 스킬 추가
- Claude Console 을 통한 버전 관리
- 커스텀 에이전트 구축을 위한 Claude Agent SDK 와 연동

언제 API 를 사용하고 언제 Claude.ai 를 사용할까?

활용 사례	최적 플랫폼
최종 사용자가 스킬과 직접 상호작용	Claude.ai / Claude Code
개발 중 수동 테스트 및 반복	Claude.ai / Claude Code
개인적, 임시 워크플로우	Claude.ai / Claude Code
프로그래매틱으로 스킬을 사용하는 애플리케이션	API
대규모 프로덕션 배포	API
자동화된 파이프라인 및 에이전트 시스템	API

오늘 적용 가능한 권장 방법

공개 레포지토리와 명확한 README, 사용 예시 스크린샷을 포함해 GitHub에 스킬을 먼저 호스팅하세요. 그런 다음 MCP 문서에 섹션을 추가해 스킬 링크를 제공하고, 두 가지를 함께 사용하는 이점과 빠른 시작 가이드를 설명하세요.

설치 가이드 예시

```
# [서비스명] 스킬 설치 방법


1. 스킬 다운로드:
   - 레포 클론: git clone https://github.com/yourcompany/skills
   - 또는 Releases에서 ZIP 다운로드


2. Claude에 설치:
   - Claude.ai > 설정 > 스킬 열기
   - "스킬 업로드" 클릭
```

```
   - 스킬 폴더(압축 파일) 선택


3. 스킬 활성화:
   - [서비스명] 스킬 토글 켜기
   - MCP 서버가 연결되어 있는지 확인


4. 테스트:
   - Claude에게 요청: "[서비스명]에서 새 프로젝트 설정해줘"
```

스킬 포지셔닝

스킬을 설명하는 방식이 사용자가 그 가치를 이해하고 실제로 시도할지 여부를 결정합니다. README, 문서, 또는 마케팅에서 스킬에 대해 작성할 때는 다음 원칙을 기억하세요.

기능이 아닌 성과에 집중하기:

```
☑ 좋음:
"ProjectHub 스킬을 사용하면 팀이 완전한 프로젝트 워크스페이스를
(페이지, 데이터베이스, 템플릿 포함) 수동 설정에 30분을 쏟는 대신
몇 초 만에 구축할 수 있습니다."


✕ 나쁨:
"ProjectHub 스킬은 YAML 프론트매터와 마크다운 명령어가 담긴 폴더로,
MCP 서버 도구를 호출합니다."
```

Chapter 5 패턴과 트러블슈팅

이 패턴들은 얼리 어답터와 내부 팀이 만든 스킬에서 도출되었습니다. 정해진 템플릿이 아니라, 효과가 입증된 일반적인 접근 방식입니다.

접근 방식 선택: 문제 우선 vs. 도구 우선

홈디포(Home Depot)처럼 생각해보세요. 문제를 가지고 들어갈 수도 있고 ("주방 캐비닛을 고쳐야 해"), 직원이 맞는 도구를 안내해줍니다. 또는 새 드릴을 고른 후 특정 작업에 어떻게 쓸지 물어볼 수도 있죠.

스킬도 같은 방식입니다:

- 문제 우선: "프로젝트 워크스페이스 설정해야 해" → 스킬이 올바른 순서로 적절한 MCP 호출을 조율합니다. 사용자는 원하는 결과를 말하고, 스킬이 도구를 처리합니다.
- 도구 우선: "Notion MCP 를 연결했어" → 스킬이 최적의 워크플로우와 모범 사례를 Claude 에게 가르칩니다. 사용자는 접근권을 가지고, 스킬이 전문성을 제공합니다.

패턴 1: 순차적 워크플로우 조율

언제 사용: 사용자가 특정 순서로 진행되는 다단계 프로세스를 필요로 할 때

```
# 워크플로우: 새 고객 온보딩


## 1단계: 계정 생성
MCP 도구 호출: create_customer
매개변수: name, email, company


## 2단계: 결제 설정
MCP 도구 호출: setup_payment_method
대기: 결제 수단 인증


## 3단계: 구독 생성
MCP 도구 호출: create_subscription
매개변수: plan_id, customer_id (1단계에서 생성된 ID)
```

```
## 4단계: 환영 이메일 발송
MCP 도구 호출: send_email
템플릿: welcome_email_template
```

핵심 기법: 명확한 단계 순서 / 단계 간 의존성 / 각 단계의 검증 / 실패 시 롤백 명령어

패턴 2: 멀티 MCP 조율

언제 사용: 워크플로우가 여러 서비스에 걸쳐 있을 때

```
# 디자인-개발 핸드오프 예시


## Phase 1: 디자인 내보내기 (Figma MCP)
1. Figma에서 디자인 에셋 내보내기
2. 디자인 사양서 생성
3. 에셋 매니페스트 생성


## Phase 2: 에셋 저장 (Drive MCP)
1. Drive에 프로젝트 폴더 생성
2. 모든 에셋 업로드
3. 공유 링크 생성


## Phase 3: 작업 생성 (Linear MCP)
1. 개발 작업 생성
2. 작업에 에셋 링크 첨부
3. 엔지니어링 팀에 할당


## Phase 4: 알림 (Slack MCP)
1. #engineering 채널에 핸드오프 요약 게시
2. 에셋 링크와 작업 참조 포함
```

패턴 3: 반복적 정제

언제 사용: 반복을 통해 결과물 품질이 향상되는 경우

```
# 반복적 보고서 생성


## 초안 작성
1. MCP를 통해 데이터 가져오기
```

```
2. 첫 번째 초안 보고서 생성
3. 임시 파일에 저장


## 품질 점검
1. 검증 스크립트 실행: scripts/check_report.py
2. 문제 파악: 누락된 섹션, 일관성 없는 포맷, 데이터 오류


## 정제 루프
1. 각 문제 해결
2. 해당 섹션 재생성
3. 재검증
4. 품질 기준 충족까지 반복


## 최종화
1. 최종 포맷 적용
2. 요약 생성
3. 최종 버전 저장
```

패턴 4: 컨텍스트 인식 도구 선택

언제 사용: 동일한 결과를 위해 상황에 따라 다른 도구가 필요한 경우

```
# 스마트 파일 저장


## 결정 트리
1. 파일 유형과 크기 확인
2. 최적 저장 위치 결정:
   - 대용량 파일 (10MB 초과): 클라우드 스토리지 MCP
   - 협업 문서: Notion/Docs MCP
   - 코드 파일: GitHub MCP
   - 임시 파일: 로컬 저장소


## 저장 실행
결정에 따라:
- 적절한 MCP 도구 호출
- 서비스별 메타데이터 적용
- 접근 링크 생성
```

```
## 사용자에게 맥락 제공
왜 그 저장소를 선택했는지 설명
```

패턴 5: 도메인 특화 인텔리전스

언제 사용: 도구 접근 이상의 전문 지식을 스킬에 추가할 때

```
# 컴플라이언스가 포함된 결제 처리


## 처리 전 (컴플라이언스 검사)
1. MCP를 통해 트랜잭션 세부 정보 가져오기
2. 컴플라이언스 규정 적용:
   - 제재 목록 확인
   - 관할권 허용 여부 검증
   - 리스크 수준 평가
3. 컴플라이언스 결정 문서화


## 처리
컴플라이언스 통과 시: MCP 도구 호출 / 사기 검사 / 트랜잭션 처리
미통과 시: 검토를 위해 플래그 / 컴플라이언스 케이스 생성


## 감사 추적
- 모든 컴플라이언스 검사 기록
- 처리 결정 기록
- 감사 보고서 생성
```

트러블슈팅

스킬이 업로드되지 않는 경우

오류: "업로드된 폴더에서 SKILL.md를 찾을 수 없습니다"

- 원인: 파일이 정확히 SKILL.md 로 명명되지 않음
- 해결: SKILL.md 로 이름 변경 (대소문자 구분). ls -la 명령으로 SKILL.md 가 표시되는지 확인

오류: "잘못된 프론트매터"

- 원인: YAML 형식 오류

```
# 잘못된 예 – 구분자 없음
```

```
name: my-skill
description: 기능을 수행합니다


# 올바른 예
---
name: my-skill
description: 기능을 수행합니다
---
```

오류: "잘못된 스킬 이름"

- 원인: 이름에 공백이나 대문자가 있음

```
# 잘못된 예
name: My Cool Skill


# 올바른 예
name: my-cool-skill
```

스킬이 트리거되지 않는 경우

증상: 스킬이 자동으로 로드되지 않음

해결책: description 필드를 수정하세요.

빠른 체크리스트:

- 너무 일반적인가? ("프로젝트를 도와줍니다"는 효과 없음)
- 사용자가 실제로 말할 트리거 문구가 포함되어 있는가?
- 관련이 있다면 파일 유형이 언급되어 있는가?

디버깅 방법: Claude에게 "[스킬 이름] 스킬을 언제 사용하나요?"라고 물어보세요. Claude가 description을 그대로 인용할 것입니다. 부족한 부분을 기반으로 수정하세요.

스킬이 너무 자주 트리거되는 경우

증상: 관련 없는 쿼리에서 스킬이 로드됨

해결책 1 - 부정적 트리거 추가:

```
description: CSV 파일의 고급 데이터 분석용. 통계 모델링, 회귀, 클러스터링에 사용하세요.
단순한 데이터 탐색에는 사용하지 마세요 (대신 data-viz 스킬 사용).
```

해결책 2 - 더 구체적으로 작성:

```
# 너무 광범위
description: 문서를 처리합니다


# 더 구체적
description: 계약서 검토를 위한 PDF 법률 문서를 처리합니다
```

MCP 연결 문제

증상: 스킬은 로드되지만 MCP 호출이 실패함

체크리스트:

5. MCP 서버 연결 확인: Claude.ai 설정 > 확장 프로그램 > [서비스명] — "연결됨" 상태여야 함
6. 인증 확인: API 키 유효 여부, 적절한 권한/스코프, OAuth 토큰 갱신
7. MCP 독립 테스트: Claude 에게 직접 MCP 호출 요청 ("[서비스명] MCP 를 사용해서 내 프로젝트 가져와")
8. 도구 이름 확인: 스킬이 올바른 MCP 도구 이름을 참조하는지 확인 (대소문자 구분)

명령어가 따르지 않는 경우

증상: 스킬은 로드되지만 Claude가 명령어를 따르지 않음

일반적인 원인:

- 명령어가 너무 장황함 — 간결하게 유지하고, 글머리 기호와 번호 목록을 사용하고, 세부 참조는 별도 파일로 이동
- 중요한 명령어가 묻혀 있음 — 상단에 중요한 명령어를 배치하고, ## 중요 또는 ## 필수 헤더 사용
- 모호한 언어 — 명확하고 구체적인 명령어 사용

```
# 나쁜 예
항목들을 제대로 검증하세요


# 좋은 예
필수: create_project 호출 전 반드시 다음을 확인:
- 프로젝트 이름이 비어있지 않음
- 최소 한 명의 팀원이 할당됨
- 시작 날짜가 과거가 아님
```

모델 "게으름" 해결책: 명시적으로 독려 추가:

```
# 성능 참고
- 이 작업을 철저하게 수행하세요
- 속도보다 품질이 중요합니다
- 검증 단계를 건너뛰지 마세요
```

대용량 컨텍스트 문제

증상: 스킬이 느리거나 응답 품질이 저하됨

원인: 스킬 콘텐츠 과대 / 너무 많은 스킬이 동시에 활성화 / 점진적 공개 없이 모든 콘텐츠가 로드됨

해결책:

9. SKILL.md 크기 최적화 — 상세 문서는 references/로 이동, SKILL.md 는 5,000 단어 이하로 유지
10. 활성화된 스킬 줄이기 — 동시에 20~50 개 이상이면 검토 필요
11. 관련 스킬들을 묶어 "스킬 팩" 고려

Chapter 6 리소스 및 참고자료

처음 스킬을 구축하는 경우, 모범 사례 가이드부터 시작한 후 필요에 따라 API 문서를 참고하세요.

공식 문서

Anthropic 리소스:

- 모범 사례 가이드
- 스킬 문서
- API 레퍼런스
- MCP 문서

블로그 게시물:

- 에이전트 스킬 소개
- 엔지니어링 블로그: 에이전트를 실전에 맞게 준비하기
- 스킬 설명
- Claude 스킬을 만드는 방법
- 스킬을 통한 프론트엔드 디자인 개선

예시 스킬

공개 스킬 레포지토리:

- GitHub: anthropics/skills
- 커스터마이즈 가능한 Anthropic 제작 스킬 포함

도구 및 유틸리티

skill-creator 스킬:

- Claude.ai 에 내장되어 있으며 Claude Code 용으로 다운로드 가능
- 설명으로부터 스킬 생성

- 검토 및 개선 제안 제공
- 사용 방법: "skill-creator 를 활용해 스킬 구축하는 것을 도와줘"

검증:

- skill-creator 로 스킬 평가 가능
- "이 스킬을 검토하고 개선 사항을 제안해줘"라고 요청하세요

지원 받기

기술적 질문:

- 일반 질문: Claude Developers Discord 커뮤니티 포럼

버그 리포트:

- GitHub Issues: anthropics/skills/issues
- 포함 정보: 스킬 이름, 오류 메시지, 재현 단계

부록 A: 빠른 체크리스트

업로드 전후에 스킬을 검증하려면 이 체크리스트를 사용하세요. 더 빠르게 시작하고 싶다면, skill-creator 스킬을 사용해 첫 번째 초안을 생성한 후 이 목록을 검토하세요.

시작 전

- 구체적인 활용 사례 2~3 개 정의
- 도구 파악 (내장 또는 MCP)
- 이 가이드와 예시 스킬 검토
- 폴더 구조 계획

개발 중

- 폴더 이름이 kebab-case 임
- SKILL.md 파일 존재 (정확한 철자)
- YAML 프론트매터에 --- 구분자 있음
- name 필드: kebab-case, 공백 없음, 대문자 없음
- description 이 '무엇을'과 '언제'를 포함함
- XML 태그 (< >) 없음
- 명령어가 명확하고 실행 가능함
- 에러 처리 포함
- 예시 제공
- 참조 파일이 명확하게 연결됨

업로드 전

- 명확한 작업에서 트리거 테스트
- 다른 표현으로 바꾼 요청에서 트리거 테스트
- 관련 없는 주제에서는 트리거되지 않는지 확인
- 기능 테스트 통과

- 도구 연동 작동 (해당하는 경우)
- .zip 파일로 압축

업로드 후

- 실제 대화에서 테스트
- 과소/과다 트리거 모니터링
- 사용자 피드백 수집
- description 및 명령어 반복 개선
- 메타데이터의 버전 업데이트

부록 B: YAML 프론트매터 참고

필수 필드

```
---
name: skill-name-in-kebab-case
description: 무엇을 하며 언제 사용하는지 설명. 구체적인 트리거 문구를 포함하세요.
---
```

모든 선택적 필드

```
name: skill-name
description: [필수 설명]
license: MIT            # 선택: 오픈소스용 라이선스
allowed-tools: "Bash(python:*) Bash(npm:*) WebFetch"  # 선택: 도구 접근 제한
metadata:               # 선택: 커스텀 필드
  author: 회사명
  version: 1.0.0
  mcp-server: server-name
  category: productivity
  tags: [project-management, automation]
  documentation: https://example.com/docs
  support: support@example.com
```

보안 참고사항

허용:

- 모든 표준 YAML 타입 (문자열, 숫자, 불리언, 목록, 객체)
- 커스텀 메타데이터 필드
- 긴 description (최대 1024 자)

금지:

- XML 꺾쇠 괄호 (< >) — 보안 제한
- YAML 에서 코드 실행 (안전한 YAML 파싱 사용)
- 'claude' 또는 'anthropic' 접두어가 포함된 스킬 이름 (예약어)

부록 C: 완전한 스킬 예시

이 가이드의 패턴을 보여주는 완전하고 프로덕션 수준의 스킬은 다음에서 확인하세요:

- 문서 스킬 — PDF, DOCX, PPTX, XLSX 생성
- 예시 스킬 — 다양한 워크플로우 패턴
- 파트너 스킬 디렉토리 — Asana, Atlassian, Canva, Figma, Sentry, Zapier 등 다양한 파트너의 스킬

이 레포지토리들은 최신 상태로 유지되며 이 가이드에서 다루는 것 이상의 추가 예시를 포함합니다. 클론하거나 수정하여 여러분의 활용 사례에 맞는 템플릿으로 활용하세요.
