EP 05 MCP & Skills

# MCP & Agent Skills

AI 에이전트에게 도구와 지식을 연결하는 두 가지 표준

AI로 업무하기 시리즈 | MCP + Agent Skills 실전편

박상돈 · AxGS Lab · 대전대학교 SW융합대학

**MCP**
**=**
**도구 연결**

**Skills**
**=**
**지식 주입**

# 문제: AI와 도구의 N×M 연결 지옥

## MCP 없이 (Before)

5개 모델 × 10개 도구
= 50개 커스텀 연동 필요

Claude ←→ GitHub (전용 코드)
Claude ←→ Slack (전용 코드)
Claude ←→ DB (전용 코드)
GPT ←→ GitHub (또 전용 코드)
GPT ←→ Slack (또 전용 코드)
...

모델이 바뀔 때마다 전부 다시 만들어야 함
도구가 추가될 때마다 전부 다시 만들어야 함

## MCP 있으면 (After)

5개 모델 + 10개 도구
= 15개 연동이면 끝

Claude ←→ MCP ←→ GitHub
GPT ←→ MCP ←→ GitHub
Gemini ←→ MCP ←→ GitHub
(같은 MCP 서버 공유!)

모델이 바뀌어도 MCP 서버는 그대로
도구가 추가되면 MCP 서버 하나만 추가

USB-C처럼 하나의 표준으로 다 연결

**MCP = AI 세계의 USB-C — 하나의 프로토콜로 어떤 모델이든 어떤 도구든 연결**

# MCP(Model Context Protocol)란?

| | |
|---|---|
| **탄생** | 2024년 11월 Anthropic이 오픈소스로 공개 |
| **채택** | OpenAI, Google DeepMind, Microsoft 모두 채택 |
| **생태계** | 2026년 4월 기준 2,300+ 공개 MCP 서버 |
| **거버넌스** | 2025.12 Linux Foundation AAIF에 기부 |
| **기반** | JSON-RPC 2.0 / stdio(로컬) + Streamable HTTP(리모트) |

# MCP 아키텍처: Host → Client → Server

## Host (AI 앱)

사용자가 직접 사용하는 앱

예: Claude Desktop, VS Code, Cursor, Claude.ai

LLM을 품고 있음
사용자의 요청을 받아서
MCP Client를 통해 도구 호출

→

## Client (MCP 클라이언트)

Host 안에 내장된 중개자

Host와 Server 사이의 통신 담당
프로토콜 규격에 맞게
요청/응답을 변환

1개 Client = 1개 Server 연결
여러 Server면 여러 Client

→

## Server (MCP 서버)

실제 도구/데이터에 접근하는 서비스

예: GitHub MCP 서버
→ GitHub API 호출

PostgreSQL MCP 서버
→ DB 쿼리 실행

Slack MCP 서버
→ 메시지 전송

**JSON-RPC 2.0으로 통신 | stdio(로컬) 또는 Streamable HTTP(리모트)**

# MCP의 3가지 프리미티브

## Tools (도구)

AI가 호출하는 함수

예시:
· query_sql → DB 쿼리 실행
· send_email → 이메일 전송
· create_issue → GitHub 이슈 생성

모델이 언제 호출할지 결정
(Model-controlled)

**AI가 결정**

## Resources (리소스)

AI가 읽는 데이터

예시:
· 파일 내용
· DB 스키마
· API 응답 데이터

URI로 식별
읽기 전용, 부작용 없음
(Application-controlled)

**앱이 결정**

## Prompts (프롬프트)

재사용 가능한 프롬프트 템플릿

예시:
· 코드 리뷰 워크플로우
· 데이터 분석 절차
· 보고서 작성 템플릿

사용자가 선택해서 호출
(User-controlled)

**사용자가 선택**

# MCP 작동 예시: "매출 데이터 보여줘"

1. 사용자가 Claude에게 "이번 달 매출 데이터 보여줘"
2. Claude(Host)가 판단: DB 조회가 필요하다
3. MCP Client가 PostgreSQL MCP Server에 요청 전달
4. MCP Server가 query_sql 도구 실행 → DB에서 데이터 가져옴
5. 결과를 JSON-RPC 응답으로 Client → Host → Claude에게 전달
6. Claude가 결과를 이해하고 사용자에게 자연어로 요약해서 답변

# MCP 서버 설정: claude_desktop_config.json

```
{
  "mcpServers": {
    "github": {
      "command": "npx",
      "args": ["-y", "@modelcontextprotocol/server-github"],
      "env": { "GITHUB_TOKEN": "ghp_xxxx" }
    },
    "postgres": {
      "command": "npx",
      "args": ["-y", "@modelcontextprotocol/server-postgres",
               "postgresql://user:pass@localhost/mydb"]
    }
  }
}
```

### 로컬 MCP 서버 (stdio)

내 컴퓨터에서 프로세스로 실행
npx로 바로 실행 가능
인증: env로 API 키 전달

### 리모트 MCP 서버 (HTTP)

클라우드에서 서비스로 실행
URL로 연결
인증: OAuth 2.1 + PKCE

# MCP 보안: 4가지 핵심 원칙

## OAuth 2.1 인증

리모트 서버 접속 시 표준 인증
PKCE로 토큰 탈취 방지

## 최소 권한 원칙

필요한 권한만 부여
DB 서버에 읽기 전용 계정 사용

## Human-in-the-loop

위험한 도구 실행 전 사용자 확인
"이 SQL 실행해도 될까요?"

## 도구 설명은 신뢰하지 않기

서버가 제공하는 도구 설명은
검증 없이 신뢰하면 안 됨

# Agent Skills란? — AI에게 전문 지식을 주입

*"아무리 똑똑한 시니어 엔지니어도 첫날에는 프로젝트 맥락이 없으면 무능하다.*
*Skills는 AI에게 그 프로젝트 맥락을 주입하는 온보딩 문서다."*

— Anthropic 공식 블로그 (2026.04)

### SKILL.md 파일

마크다운 파일 하나가 곧 하나의 스킬
YAML frontmatter + 지시사항으로 구성

### 온디맨드 로딩

필요할 때만 읽어서 컨텍스트 낭비 없음
Progressive Disclosure 원칙

### 크로스 플랫폼

Claude Code, Claude.ai, Cursor,
Gemini CLI, OpenCode 등에서 동작

# SKILL.md 파일 구조: 실제 예시

```
---
name: explain-code
description: >
  코드를 시각적 다이어그램과 비유로
  설명합니다. "이 코드 어떻게 동작해?"
  같은 질문에 자동 로딩됩니다.
---

# explain-code

## 코드 설명 시 항상 포함할 것

1. 일상 비유로 시작하기
   코드를 일상생활에 비유

2. 다이어그램 그리기
   ASCII 아트로 흐름/구조 표시

3. 단계별 워크스루
   코드 실행 순서대로 설명

4. 흔한 실수 하나 짚기
   초보자가 빠지기 쉬운 함정
```

## YAML frontmatter

name → /slash-command 이름
description → AI가 언제 이 스킬을 로딩할지 판단하는 기준

자동 로딩: description이 매치되면
수동 호출: /explain-code 입력

## Markdown 지시사항

AI가 이 스킬을 실행할 때 따라야 할 구체적 지침

지시사항이 구체적일수록 결과물 품질이 올라감

하네스의 AGENTS.md와 같은 원리!

# Progressive Disclosure: 핵심 설계 원칙

**1**

## Level 1: 이름 + 설명

시스템 프롬프트에 사전 로딩
모든 스킬의 name/description만
→ 어떤 스킬이 있는지만 파악

수십 바이트

**2**

## Level 2: SKILL.md 전체

관련 작업이 들어오면
bash로 SKILL.md를 읽음
→ 핵심 지시사항 로딩

수백 바이트~수 KB

**3**

## Level 3: 참조 파일

SKILL.md가 references/를 가리키면
필요한 파일만 추가로 읽음
→ 상세 문서, 스크립트 등

필요한 만큼만

**잘 정리된 매뉴얼처럼: 목차 → 해당 챕터 → 부록. 처음부터 다 읽지 않는다.**

# 실전 예시: Anthropic PDF Skill의 작동 과정

1. 사용자: "이 PDF 양식 채워줘"
2. Claude: PDF 스킬 발견 → bash로 pdf/SKILL.md 읽기
3. SKILL.md에서 "양식 채우기는 forms.md 참조" 발견
4. Claude: forms.md도 읽기 (필요할 때만 추가 로딩)
5. SKILL.md에 포함된 Python 스크립트 실행
   → PDF의 폼 필드 추출 (스크립트 코드 자체는 컨텍스트에 안 들어감!)
6. 추출된 필드 정보로 양식을 채우고 결과 PDF 반환

# Skills 설치 위치와 설치 방법

## 글로벌 스킬 (모든 프로젝트)

~/.claude/skills/스킬이름/SKILL.md

모든 프로젝트에서 사용 가능
개인 워크플로우 자동화에 적합

예시:
~/.claude/skills/explain-code/SKILL.md
~/.claude/skills/git-commit/SKILL.md

## 프로젝트 스킬 (이 프로젝트만)

프로젝트/.claude/skills/스킬이름/SKILL.md

이 프로젝트에서만 사용 가능
팀과 Git으로 공유 가능

예시:
my-app/.claude/skills/deploy/SKILL.md
my-app/.claude/skills/test/SKILL.md

## 설치 방법

```
# Anthropic 공식 스킬
npx skills add anthropics/skills --skill frontend-design

# 커뮤니티 스킬 (Antigravity Awesome Skills – 1,234+ 스킬)
npx antigravity-awesome-skills --claude

# 직접 만들기
mkdir -p ~/.claude/skills/my-skill && vi ~/.claude/skills/my-skill/SKILL.md
```

# MCP vs Skills: 도구 vs 지식

## MCP = 도구 (Hands)

AI에게 손을 쥐여주는 것

"뭘 할 수 있게 해주는가?"

· DB를 조회할 수 있다
· 이메일을 보낼 수 있다
· GitHub에 코드를 올릴 수 있다
· Slack에 메시지를 보낼 수 있다

실시간 연결, 양방향 통신
JSON-RPC 2.0 프로토콜

## Skills = 지식 (Brain)

AI에게 지식을 주입하는 것

"어떻게 해야 하는지 아는가?"

· PDF 양식을 이렇게 채워야 한다
· 프론트엔드는 이 패턴으로
· 코드 리뷰는 이 절차로
· 배포는 이 순서로

온디맨드 로딩, .md 파일 기반
Progressive Disclosure

**MCP가 도구를 연결하고, Skills가 사용법을 알려준다.**
**예: Sentry MCP 서버(도구 접근) + sentry-code-review Skill(리뷰 워크플로우)**

# 하네스 엔지니어링과의 연결

| 하네스 3기둥 | MCP의 역할 | Skills의 역할 |
| --- | --- | --- |
| **컨텍스트 파일 (AGENTS.md)** | — | SKILL.md가 정확히 이 역할<br>프로젝트별 지식과 규칙 정의 |
| **자동 강제 시스템 (린터/훅)** | 도구 호출 전 사용자 확인<br>(Human-in-the-loop) | 지시사항으로 패턴 강제<br>(AI slop 금지 등) |
| **가비지 컬렉션 (청소 에이전트)** | — | Progressive Disclosure로<br>불필요한 정보 로딩 방지 |
| **멱등성 (모델 무관 일관성)** | 모델 무관 도구 접근<br>(USB-C 표준) | 모델 무관 지식 주입<br>(크로스 플랫폼) |

# 핵심 정리

1

## MCP = AI의 USB-C

하나의 프로토콜로 어떤 모델이든 어떤 도구든 연결

2

## Skills = AI의 온보딩 문서

SKILL.md 하나로 전문 지식을 주입 — 크로스 플랫폼

3

## MCP는 도구, Skills는 지식

MCP가 손을 쥐여주고, Skills가 사용법을 알려준다

4

## Progressive Disclosure로 컨텍스트 절약

필요한 것만, 필요할 때만 로딩 — 1000개 스킬도 OK

5

## 하네스 3기둥의 확장

컨텍스트 파일 + 자동 강제 + 가비지 컬렉션의 표준화된 버전