작동 원리 상세 설명

개요

이 프로젝트는 **Claude Code 슬래시 커맨드 시스템**을 활용하여 비개발자도 쉽게 사용할 수 있는 도구 배포 방식을 구현했습니다.

전체 구조

```
1. 도구 제작자 (개발자)
   ↓
2. GitHub에 업로드
   ↓
3. 사용자가 명령어 설명서 다운로드
   ↓
4. Claude Code가 설명서를 읽고 자동 실행
   ↓
5. 실제 도구가 작동
```

작동 방식 (단계별)

1단계: 도구 제작

A. 실제 프로그램 개발 (`md2pdf` CLI 도구)

- TypeScript로 마크다운→PDF 변환 프로그램 작성
- `npm link`로 전역 명령어로 등록
- GitHub에 업로드

핵심 파일:

- `src/md2pdf.ts` - 실제 변환 로직
- `package.json` - npm 패키지 설정
- `bin/md2pdf.js` - CLI 진입점

B. Claude용 설명서 작성 (`.claude/commands/md2pdf.md`)

- Claude Code가 읽을 수 있는 **지시서**
- "사용자가 `/md2pdf` 를 입력하면 이렇게 행동해라"는 매뉴얼
- **이 파일이 핵심입니다!**

2단계: 사용자가 설치 (한 줄 명령어)

```
mkdir -p .claude/commands && curl -o .claude/commands/md2pdf.md htt
```

이 명령어가 하는 일:

1. `.claude/commands` 폴더 생성
2. GitHub에서 설명서 파일만 다운로드
3. **실제 프로그램은 아직 설치되지 않음!**

3단계: 사용자가 `/md2pdf README.md` 입력

Claude Code가 하는 일:

1. `.claude/commands/` 폴더에서 `md2pdf.md` 파일 발견
2. 파일을 읽어서 "아, 이렇게 해야 하는구나" 이해
3. 설명서에 적힌 대로 **자동으로 실행:**
   - md2pdf 명령어가 설치되어 있는지 확인 (`which md2pdf`)
   - 없으면? → GitHub 클론 → npm install → npm build → npm link
   - 있으면? → 바로 md2pdf 실행
4. 파일 검색 및 PDF 변환 실행
5. 결과를 사용자에게 알림

4단계: PDF 생성 완료!

핵심 개념

`.claude/commands/md2pdf.md` 의 역할

이 파일은 **Claude Code에게 주는 각본(스크립트)**입니다:

"사용자가 /md2pdf를 입력하면:
1. 먼저 설치됐는지 확인해
2. 안 됐으면 GitHub에서 클론해서 설치해
3. 파일 찾아서
4. md2pdf 명령어 실행해
5. 결과 알려줘"

왜 이 방식이 혁신적인가?

기존 방식 (복잡 - 개발자만 가능)

```
# 사용자가 직접 해야 함
git clone https://github.com/daht-mad/md2pdf.git
cd md2pdf
npm install
npm run build
npm link
md2pdf README.md
```

문제점:

- Git, npm, 터미널 명령어를 알아야 함
- 여러 단계를 순서대로 실행해야 함
- 에러 발생 시 디버깅 능력 필요

새로운 방식 (간단 - 비개발자도 가능)

```
# 1. 설치 (한 번만, 복사-붙여넣기)
curl -o .claude/commands/md2pdf.md https://raw.githubusercontent.co

# 2. 사용 (Claude Code에서, 자연어처럼)
/md2pdf README.md
```

장점:

- Claude가 중간에서 모든 복잡한 작업을 대신 처리
- VSCode 안에서 자연어처럼 사용
- 에러 처리도 Claude가 알아서 해결

기술적 구조

구성 요소

1. CLI 도구 (`src/md2pdf.ts`)

- 실제 변환 로직
- npm 패키지로 배포 가능
- 독립적으로도 사용 가능

주요 기능:

- 파일 재귀 검색 (BFS 알고리즘)
- 중복 파일 선택 UI
- Markdown → HTML → PDF 변환
- 한글 폰트 처리

2. Claude Code 확장 모듈 (`.claude/commands/md2pdf.md`)

- Claude Code의 "플러그인" 같은 역할
- Prompt Engineering으로 작성된 **자동화 스크립트**
- Claude가 이 파일을 읽고 **코딩 없이도** 복잡한 작업 수행

핵심 내용:

- 설치 확인 로직
- 자동 설치 프로세스
- 파일 검색 및 실행 로직
- 에러 처리 가이드

이 방식의 장점

사용자 관점

- ✅ **비개발자 친화적**: 터미널 명령어 몰라도 됨
- ✅ **자동 설치**: 처음 실행 시 알아서 설치
- ✅ **VSCode 통합**: 익숙한 환경에서 사용
- ✅ **자연어 인터페이스**: `/md2pdf README.md` 처럼 직관적

개발자 관점

- ✅ **팀 공유 쉬움**: 파일 하나만 공유하면 됨 (Git 커밋)
- ✅ **업데이트 쉬움**: 설명서만 업데이트하면 됨
- ✅ **의존성 관리 자동화**: npm, git 설정 불필요
- ✅ **확장성**: 다른 도구에도 적용 가능

배포자 관점

- ✅ **중앙 집중 관리**: GitHub 저장소 하나로 관리
- ✅ **버전 관리**: Git으로 자연스럽게 버전 관리
- ✅ **사용자 지원 간소화**: Claude가 대부분의 문제 해결
- ✅ **문서화**: 코드와 사용법이 한 곳에

메타포: Claude Code용 앱스토어

이 시스템은 사실상 **"Claude Code용 앱스토어"**입니다:

요소	역할
앱스토어	GitHub 저장소
앱	md2pdf 도구
앱 설치 링크	`.claude/commands/md2pdf.md`
앱 실행	`/md2pdf` 슬래시 커맨드
앱 업데이트	`git pull` (자동)

확장 가능성

이 패턴을 다른 도구에도 적용할 수 있습니다:

예시

- `/db-migrate` - 데이터베이스 마이그레이션 도구
- `/deploy` - 배포 자동화 도구
- `/test-coverage` - 테스트 커버리지 리포트
- `/screenshot` - 웹사이트 스크린샷 캡처
- `/api-docs` - API 문서 자동 생성

필요한 것

1. 실제 기능을 하는 CLI 도구
2. `.claude/commands/{도구명}.md` 설명서 파일
3. GitHub 저장소

결론

이 시스템은 **AI 에이전트(Claude)를 활용한 도구 배포 및 자동화**의 새로운 패러다임입니다.

핵심 아이디어:

- 복잡한 설치/실행 과정을 자연어 지시서로 문서화
- AI가 문서를 읽고 자동으로 실행
- 사용자는 간단한 명령어만 입력

결과:

- 개발자 도구를 비개발자도 사용 가능
- 팀 협업 시 설정 공유 자동화
- 유지보수 부담 최소화

이것이 **md2pdf 프로젝트의 진짜 혁신**입니다! 🎉