요즘 내가 고민하는 것

"비개발자가 도구를 만들고, 공유하고, 서로의 기여로 성장하는 생태계"

---

1. 출발점: 어쩌다 만들어버린 기능들

클로드코드로 일하다 보면, 그때그때 엄청난 기능들을 만들어준다.

예를 들어,

- "이 마크다운 파일 PDF로 변환해줘" → 변환 스크립트 만들어줌
- "이미지 용량 줄여줘" → 압축 로직 만들어줌
- "이 엑셀 데이터 정리해줘" → 자동화 코드 만들어줌

근데 문제는 **이때 만든 기능을 재사용하기가 어렵다**는 것. 다음에 비슷한 작업할 때 또 처음부터 설명해야 한다.

그래서 생각했다.

> "어쩌다 만들어버린 기능을 **도구로 모듈화**해서, 나도 쓰고 남도 쓰게 하면 어떨까?"

더 나아가서,

- 우리 팀원들이 각자 만든 기능을 공동 저장소에 모아놓고 갖다 쓰게 하면?
- 비개발자 누구나 한 줄 명령어로 서로의 도구를 갖다 쓸 수 있으면?
- 그러려면 동일한 방식으로 도구를 만드는 가이드가 있으면 되겠다
- **그러면 "도구를 만드는 도구"를 만들면 되겠다!**

---

2. 내가 만든 것: 도구를 만드는 도구

핵심 작동 방식

Claude Code에는 **슬래시 커맨드**라는 기능이 있다. `.claude/commands/` 폴더에 마크다운 파일을 넣어두면, `/파일명`으로 실행할 수 있다.

이걸 활용해서 **"도구를 만드는 도구"**를 만들었다.

```
/create-tool
```

이 한 줄을 입력하면, Claude가 다음을 자동으로 처리한다:

1. **대화로 요구사항 파악**: "어떤 기능을 만들고 싶으신가요?"
2. **코드 자동 생성**: Node.js 기반 CLI 도구 작성
3. **테스트 실행**: 만든 코드가 제대로 동작하는지 확인
4. **GitHub 저장소 생성**: 자동으로 레포지토리 만들고 푸시
5. **npm 배포**: 전 세계 누구나 설치할 수 있게 배포
6. **설치 명령어 제공**: curl 한 줄로 설치 가능한 형태로

실제 사용 예시

만드는 사람 (비개발자)

```
나: /create-tool
Claude: 어떤 기능을 만들고 싶으신가요?
나: 마크다운 파일을 예쁜 PDF로 변환하고 싶어요. 한글 폰트도 지원하고.

[3분 후]

Claude: ✅ md2pdf 도구 생성 완료!
- GitHub: https://github.com/username/md2pdf
- npm: https://npmjs.com/package/md2pdf
- 설치 명령어: curl -o .claude/commands/md2pdf.md https://raw...
```

쓰는 사람 (비개발자)

```
# 설치 (복사-붙여넣기 한 번)
mkdir -p .claude/commands && curl -o .claude/commands/md2pdf.md
https://raw.githubusercontent.com/...

# 사용
/md2pdf README.md
```

왜 이게 가능한가?

Claude Code의 구조적 특성을 활용했다.

요소	역할
슬래시 커맨드	복잡한 작업을 한 단어로 실행 (`/create-tool`)
마크다운 프롬프트	도구의 "설계도"를 자연어로 작성
Claude의 코드 생성	설계도를 보고 실제 코드 작성
GitHub + npm 연동	자동 배포까지 처리

핵심 원리: 상위 개념 만들기

필요한 기능 하나를 만들고 → 그걸 만드는 기능을 상위 개념으로 또 만든다.

```
md2pdf (마크다운→PDF 변환 도구)
    ↑ 이걸 만들어주는
create-tool (도구를 만드는 도구)
    ↑ 이걸 만들어주는
??? (도구를 만드는 도구를 만드는 도구)
```

이렇게 추상화 레벨을 올리면, **Claude가 "더 오래" 일하게 할 수 있다.** 한 번 지시로 더 많은 것을 자동화할 수 있다.

---

3. 근데, 여기서 멈추면 안 되는 이유

지금 기존 도구 저장소들의 문제

서비스	문제점
GitHub	비개발자는 clone, fork, PR이 뭔지 모름
MCP 저장소 (smithery)	설치는 쉬워졌지만, 만드는 건 여전히 개발자만
Claude Code Skills	스킬 만들고 공유하는 건 개발자뿐
npm	패키지 배포? 비개발자에겐 외계어

세상의 90%는 비개발자인데, 도구 생태계는 전부 개발자 친화적이다.

오픈소스라고 해놔도 비개발자는:

- 어떻게 설치하는지 모름
- 어떻게 사용하는지 모름
- 당연히 어떻게 만드는지도 모름

비개발자가 겪는 현실

1. **만들 수 없다**: AI로 뭔가 만들어도, 재사용 가능한 형태로 모듈화 못함
2. **배포할 수 없다**: Git, npm, CLI 명령어 모름
3. **역량을 증명할 수 없다**: "저 이런 거 만들었어요"를 보여줄 방법 없음
4. **협업 기회가 없다**: 개발자와 만날 접점 자체가 없음

---

4. 시대적 흐름: 왜 지금 이게 필요한가

생산의 민주화 역사

단계	시대	의미	대표 기술
1단계	2010년대 초	누구나 제품 생산자	3D 프린터
2단계	2010년대 중반	누구나 콘텐츠 생산자	유튜브, 브런치, 틱톡
3단계	2020년대 초	누구나 서비스 생산자	노코드(Notion, Zapier), AI 코딩
4단계	**지금**	모두가 서로의 생산자	초개인화 AI 시대

4단계가 의미하는 것

"생산자가 소비자보다 많아지는 시대"

유튜브를 생각해보자.

- 처음엔 콘텐츠 만드는 사람이 적고, 보는 사람이 많았다
- 지금은? 거의 모든 사람이 유튜브에 영상 하나쯤은 올려봤다
- 공급이 폭발적으로 늘어남 → 조회수로 돈 벌기 어려워짐

이게 "서비스" 영역에서도 일어나고 있다.

AI 코딩 도구 덕분에:

- 아이디어만 있으면 누구나 서비스를 만들 수 있다

- "이거 나도 AI한테 만들어달라고 하면 되는데, 왜 돈 주고 써?"
- 남이 만든 거 쓰기보다 **내가 만들고 소유**하고 싶어진다

핵심 질문의 전환

예전 질문:

> "남이 내 걸 어떻게 쓰게 할 것인가?" (사용자 늘리기, 구독료, 광고...)

새로운 질문:

> **"남이 어떻게 쉽게 베껴가게 할 것인가?" "그 기여를 어떻게 자산화할 것인가?"**

공급 과잉 시대에는 "팔아서" 돈 버는 게 아니라, "기여"로 돈 버는 구조가 필요하다.

smithery.ai의 변화가 보여주는 것

smithery.ai는 MCP 서버 저장소인데, 최근에 흥미로운 기능이 추가됐다.

> "채팅으로 스킬 만들기"

코드 먼저가 아니라 **대화 먼저**로 도구를 만들 수 있게 한 것. 업계 자체가 이 방향으로 가고 있다는 증거다.

AI 시대의 학습 방식 변화

앞으로 학습은 대부분 **AI와 대화하며 커스텀 방식**으로 하게 될 것.

- 별도 앱을 만드는 게 아니라
- AI에게 구조와 지침을 주고
- 이 인프라를 활용한 교육으로 가는 방향

전제 자체가 바뀐다:

- ~~"사람이 한다"~~
- → **"에이전트 시킨다"**

그러면 남는 질문:

> **"AI가 도와주지 못하는 지점을 어떻게 사람들끼리 서로 돕게끔 만들 것인가?"**

---

5. 내가 그리는 그림

비개발자를 위한 오픈소스 도구 생태계

만들고 → 공유하고 → 베끼고 → 개선하고 → 다시 공유

```
[나]                          [남]
도구 만듦
     → 오픈소스 배포 →
                              클론해서 사용
                              내 상황에 맞게 개선
                         ← 개선된 버전 재배포 ←
나도 그 개선 버전 씀
```

핵심 가치

"비개발자가 자신의 역량을 증명할 수 있는 곳"

행동	결과
도구를 만들어 배포하면	**포트폴리오**가 된다
사용량/클론 수가 쌓이면	**신뢰도**가 된다
활동 이력이 쌓이면	**역량 증명**이 된다
역량이 쌓이면	**협업 기회 + 수익**으로 연결된다

단계별 기능

1단계: 도구 배포/설치 시스템

- 플랫폼의 기반
- curl 한 줄 설치
- 클라이언트별 선택 (Claude Code, 안티그래비티, 커서 등)

2단계: 마이페이지

- 내가 만든 도구들 모아보기
- 사용량/클론 통계
- 기여의 자산화를 눈에 보이게

3단계: 팀페이지

- 우리 팀이 만들고 함께 공유할 도구들
- 비개발자 조직에서 "어쩌다 만들어버린 것들"을 한 곳에

---

6. 수익 모델: "나만의 개발자 찾기"

왜 개발자가 필요한가?

비개발자가 AI로 **80%는 만들 수 있다.** 근데 나머지 20%가 문제다:

- 보안
- 인프라 (DB, 서버)
- 최적화
- 트러블슈팅

이건 아직 전문가가 필요하다.

작동 방식

참여자	얻는 것
비개발자	코드 리뷰, 기능 개선, 인프라 구축 도움
개발자	수익 + "비개발자 협업 전문가" 브랜딩

비개발자 참여 자격

- 플랫폼 가입: 누구나 자유롭게
- 활동: 도구 만들고, 배포하고, 갖다 쓰고
- 역량 축적: 활동 자체가 포트폴리오 + 역량 증명
- **나만의 개발자 찾기 참여**: 축적된 역량으로 자격 획득

→ 진입장벽 낮음 + 개발자에겐 검증된 비개발자만 연결

개발자의 역할 재정의

기존:

> "코드 짜주는 사람"

새로운 역할:

> **"비개발자가 빛나게 인프라 깔아주는 사람"**

- 비개발자의 강점(기획, 디자인, 아이디어)을 최대한 부각
- 개발자는 뒤에서 인프라를 깔아주는 서포터

개발자에게도 메리트가 있다

즉각적 보상: 현금 수익

장기적 가치: 미래 경쟁력 확보

- AI로 비개발자도 개발하는 시대가 온다
- "코드만 잘 짜는 개발자"는 점점 대체된다
- **"비개발자의 아이디어를 현실로 만들어주는 개발자"**는 희소해진다
- 이 플랫폼이 그런 개발자를 **증명하는 곳**이 된다

---

7. 기여의 자산화: 도토리 시스템

오픈소스로 하되, **기여를 측정**한다.

활동	도토리
도구 배포	획득
내 도구가 클론됨	획득
클론 후 개선해서 재배포	획득
남의 도구 사용	지불
커피챗 요청	지불

다단계 형태로 기여 순환

```
최초 기여자 (도구 원작자)
      ↓ 일정 비율 배분
클론 기여자 (개선해서 재배포)
      ↓ 일정 비율 배분
```

```
클론클론 기여자 (또 개선해서 재배포)
    ...
```

하위에서 발생한 기여도 상위로 일정 부분 올라감. **기여가 기여를 낳는 구조.**

MVP에서는

도토리 시스템 없이 시작.

- 클론 횟수, 사용량 등 **데이터만 먼저 축적**
- 커뮤니티가 성장하면 도토리 시스템 도입
- 기존 데이터를 도토리로 전환

---

8. 현재 고민 지점

1. 범용성 문제

클로드코드에서만 쓰면 시장이 좁다.

클라이언트	장점
Claude Code	슬래시 커맨드, 에이전트(병렬실행)
안티그래비티	비개발자가 무료로 시작하기 좋음
커서	개발자들이 많이 씀

→ **설치 명령어를 클라이언트별로 선택**할 수 있게 해야 함 (smithery가 MCP 설치할 때 클라이언트 선택하게 하는 것처럼)

2. AI가 해결 못하는 건 뭔가?

- 사람이 직접 해야 하는 부분은?
- 사람들끼리만 가능한 건?
  - 팀코칭?
  - 휴먼터치?
  - 서로돕기?

→ **이 지점이 플랫폼의 핵심 차별점**이 될 수 있다

3. 닭과 달걀

비개발자와 개발자 양쪽을 동시에 모아야 한다.

전략:

1. **비개발자 커뮤니티 먼저 활성화** (다오랩 등 이미 속한 곳에서 시작)
2. 개발자는 **수익 기회** 보고 유입되게
3. 행사/교육과 연계하여 인식 확산

4. 시대를 앞서감

AI 코딩 도구 자체가 아직 대중화 안 됨.

→ 얼리어답터 커뮤니티부터 시작 → 그들이 만드는 도구와 성공 사례가 쌓이면 자연스럽게 확산

---

9. 한 줄 요약

> **"비개발자가 도구를 만들고, 오픈소스로 공유하고, 그 기여가 역량 증명과 협업 기회로 이어지는 생태계를 만들고 싶다."**

---

10. 왜 이게 중요한가

시대의 방향

```
누구나 제품 생산자 (3D 프린터)
    ↓
누구나 콘텐츠 생산자 (유튜브, 브런치)
    ↓
누구나 서비스 생산자 (노코드, AI)
    ↓
모두가 서로의 생산자 ← 우리는 여기를 연다
```

남는 질문

AI가 대부분을 해결해주는 시대에, **사람이 해야 할 것**은 무엇인가?

내 답:

- **서로의 기여를 연결하는 것**
- **기여를 통해 서로의 성장을 촉진하는 것**

이 생태계가 그 인프라가 되면 좋겠다.

참고: 비슷한 시도들

커서맛피아님 (greatSumini)

- 커맨드, 스킬, 에이전트 만드는 프롬프트 오픈소스 공개
- monet.design - 웹사이트 컴포넌트 추출/공유 플랫폼
- 누구나 기여 가능 → "오픈소스 장점이 사람들이 알아서 수정해준다"

smithery.ai

- MCP 서버 저장소
- 최근 "채팅으로 스킬 만들기" 기능 추가
- 코드 먼저 → 대화 먼저로 전환

업계 전체가 이 방향으로 가고 있다.