

AI 1인 사업 바이블

Claude Code 실전부터 디지털 상품 수익화,
1인 사업 자동화까지

237

쪽

28

개 장

67

프롬프트

155

체크리스트

12

템플릿

차례

들어가며	4
------	---

PART 1 · Claude Code 실전 — 기초 체력

01	Claude Code가 바꾸는 것: 챗봇과 에이전트의 차이	10
02	설치부터 첫 결과물까지 30분	17
03	권한과 안전장치	24
04	CLAUDE.md 설계법	33
05	컨텍스트·모델·비용 관리	39
06	일 맡기는 법: 계획 → 실행 → 검증 루프	45
07	막혔을 때 복구 루틴	52

PART 2 · Claude Code 고급

08	커스텀 명령과 스킴	59
09	서브에이전트로 역할 나누기	66
10	훅: 사람이 깜빡해도 지켜지는 규칙	73
11	MCP로 외부 도구 연결	80
12	자동 실행: 헤드리스·GitHub·예약	87
13	설정·플러그인·팀 공유 정리	95
14	실전 프로젝트 3개 따라 하기	101

PART 3 · 디지털 상품 수익화

15	팔리는 디지털 상품 고르기	109
16	AI 제작 파이프라인	117
17	상세페이지·썸네일·미리보기·영상	125
18	가격 전략	132
19	판매 채널 비교와 세팅	138

20	스레드로 파는 법	145
21	고객·법·신뢰	152

PART 4 · 1인 사업 자동화

22	자동화 지도 그리기	162
23	콘텐츠 자동화	169
24	판매·고객 응대 자동화	175
25	숫자 자동 집계	182
26	사이트 운영 자동화	188
27	AI 에이전트에게 반복 업무 맡기기	195
28	주간 운영 루틴과 90일 로드맵	204
	부록 A. 체크리스트 155	211
	부록 B. 이 책의 프롬프트 한눈에	224
	부록 C. 참고 자료	229

들어가며

■ 112건과 0건 사이에서 배운 것

저는 코드를 배운 적 없는 비개발자입니다. AICONTENTLAB이라는 이름으로 전자책 플랫폼 래피드에 15권 넘게 전자책을 올려 팔고 있습니다. 숫자부터 솔직하게 말씀드리겠습니다. 「AI가 만든 사이트, 다시 디자인합니다」는 **112건** 팔렸고, 「부업의 종류와 실행과정」은 13건 팔렸습니다. 나머지 책은 대부분 **0~3건**입니다. 책을 많이 만들었지만, 대부분은 안 팔렸습니다.

같은 사람이 같은 도구로 만들었는데 왜 한 권은 112건이고 다른 책은 0건이었을까요. 이 질문이 이 책의 출발점입니다. 만드는 속도는 AI 덕분에 이미 충분했습니다. 차이는 **무엇을 만들지 고르는 법, 검증하는 법, 파는 과정을 반복 가능한 구조로 만드는 법**에서 났습니다. 구체적인 비교는 15장에서 합니다.

그사이 Claude Code와 함께 유료 구독 사이트 두 개도 처음부터 만들었습니다. 하루 1분 질문에 답을 쓰는 **mezip.kr**, 매주 추리 사건을 푸는 **주간미제**입니다. Cloudflare Workers와 D1 위에서 돌아가고 회원가입, 로그인, 비밀번호 재설정, 관리자 화면, 약관, 탈퇴까지 갖췄습니다. 결제는 계좌이체를 확인한 뒤 관리자가 승인하는 수동 멤버십입니다. 자동 테스트는 300개가 넘고, 별도 AI에게 해커 역할을 맡긴 보안 검토크를 두 번 받아 문제 16건을 찾아 고쳤습니다. 두 사이트는 아직 공개할 만한 매출 숫자가 없고, PG 정기결제도 아직 붙이지 않았습니다. 이 책에서 두 사이트는 "이렇게 만들고 이렇게 검증했다"는 근거로만 씁니다.

운영도 Claude에게 맡겨 왔습니다. 원고 PDF 조판, 상세페이지 이미지, 홍보 영상 렌더링, 래피드 상품 등록, 스레드 예약 게시까지입니다. 그러면서 "스레드 예약은 25개까지만 걸린다", "래피드 사다리 가격은 자동으로 오르지 않는다", "비밀번호와 결제 버튼은 사람이 눌러야 한다" 같은 제약을 몸으로 배웠습니다. 이 책은 그 전 과정을 한 줄로 켜 기록입니다.

이 책이 기존 책과 다른 점

이 책은 제가 주제별로 낸 작은 책들을 하나로 꿰는 **상위판**입니다. 겹치는 내용은 한 줄로 짚고 더 깊이 들어갑니다.

앞선 책에서 한 것	이 책에서 더 하는 것
「한도에 안 걸리는 AI 코딩」: 토큰·컨텍스트 절약 습관	권한·계획 모드·체크포인트까지 묶은 작업 운영 체계 (PART 1)
「AI가 망가뜨린 코드, 30분 안에 되살리기」: 복구 순서	루프 신호 판별, 인수인계 메모, 배포 사고 대응 순서 (07장)
「AI로 돈 받는 사이트 만들기」: 사이트 한 개	스킬·서브에이전트·훅·MCP로 나만의 개발팀 만들기 (PART 2)
「AI로 상세페이지 하루 컷」: 상세페이지 구조	상품 선정 → 가격 → 채널 → 법까지 판매 전 과정 (PART 3)
「노트북을 꺼도 AI가 일한다」: 봇 5종	자동화 지도, 숫자 집계, 주간 루틴과 90일 로드맵 (PART 4)

4개 파트 지도

파트	답하는 질문	끝나면 손에 남는 것
PART 1. Claude Code 실전 — 기초 체력 (01~07장)	AI에게 일을 안전하게, 정확하게 맡기려면?	설치된 환경, 권한 설정 파일, CLAUDE.md, 작업 카드, 복구 루틴
PART 2. Claude Code 고급 — 나만의 개발팀 (08~14장)	반복 작업을 어떻게 명령· 스킬·자동 실행으로 바꾸나?	커스텀 스킬, 리뷰어 에이전트, 훅, 실전 프로젝트 3개
PART 3. 디지털 상품 수익화 (15~21장)	무엇을 만들어 어디서 얼마에 어떻게 파나?	상품 기획서, 상세페이지, 가격표, 채널 세팅, 법 점검표
PART 4. 1인 사업 자동화 (22~28장)	내가 없어도 돌아가는 구조는 어떻게 만드나?	자동화 지도, 콘텐츠·응대·집계 파이프라인, 주간 루틴

독자 유형별 읽는 순서

지금 내 상황	읽는 순서	오늘 할 첫 행동
Claude Code를 아직 설치하지 않았다	01 → 02 → 03 → 04 → 06 → 나머지 순서대로	02장 30분 실습으로 첫 결과물 만들기
바이브코딩은 해 봤는데 자꾸 망가진다	03 → 06 → 07 → PART 2	03장 settings 파일과 Git 세이브 포인트 만들기
만들 줄은 아는데 안 팔린다	15 → 17 → 18 → 19 → 20 → 21	15장 수요 확인 5가지 중 2가지 해 보기
이미 팔고 있고 시간이 모자란다	22 → 27 → 25 → 28	22장 업무 목록과 점수표 작성

준비물과 비용 (2026년 10월 확인 기준)

준비물	꼭 필요한가	비용	메모
Claude 유료 요금제 (Pro 이상) 또는 Claude Console 계정	필수	Pro 월 20달러부터 (세금 별도)	무료 요금제에는 Claude Code가 포함되지 않습니다. 고르는 법은 02장
컴퓨터 (macOS 13 이상 / Windows 10 1809 이상 등, RAM 4GB 이상)	필수	보유 장비	공식 문서의 시스템 요구사항 기준
Git	강력 권장	무료	03장부터 안전망으로 씁니다
코드 편집기 (VS Code 등)	권장	무료	결과물을 눈으로 확인하는 용도
Node.js	06장 실습부터	무료	점검 스크립트·테스트 실행용
Cloudflare 계정	PART 2 실전부터	무료 플랜으로 시작 가능	무료 한도는 26장에서 공식 기준으로 정리합니다
판매 플랫폼 계정 (래피드, 그로블 등)	PART 3부터	가입 무료, 판매 시 수수료	수수료·정산 규칙은 19장
도메인	선택	업체·주소마다 다름	없어도 시작할 수 있습니다

요금은 바뀝니다. 결제 전에 claude.com/pricing에서 다시 확인하세요.

이 책이 약속하지 않는 것

- **수익을 보장하지 않습니다.** 제 책도 대부분 안 팔렸습니다. 이 책은 실패 확률을 줄이는 절차를 드립니다.
- **기능이 그대로라고 약속하지 않습니다.** Claude Code는 업데이트가 잦습니다. 이 책의 명령·설정은 2026년 10월에 공식 문서로 확인한 것이고, 출처는 부록에 모았습니다.

화면이 다르면 공식 문서를 먼저 보세요.

- **법률·세무 자문이 아닙니다.** 21장의 법·세금 내용은 확인한 범위의 안내이고, 판단은 공식 기관과 전문가에게 확인하세요.

이 책을 쓰는 법

- **복불 프롬프트**는 [대괄호] 부분만 내 상황으로 바꿔 그대로 붙여 넣으면 됩니다.
- **"예시:"**라고 적힌 사례는 설명을 위한 가상 예시입니다. 표시가 없는 경험담은 모두 제가 실제로 겪은 일입니다.
- 장마다 끝에 **체크리스트**가 있습니다. 읽기만 하지 말고 체크박스를 하나씩 지우세요. 부록에 전체 체크리스트를 모아 두었습니다.

> 복불 프롬프트

[대괄호]만 바꾸기

나는 이 책 「AI 1인 사업 바이블」을 읽고 실행하려는 사람이야. 내 상황에 맞는 4주 실행 계획을 세워줘.

내 상황:

- 코딩 경험: [전혀 없음 / 바이트코딩 몇 번 / 사이트 하나 만들어 봄]
- 지금 있는 것: [Claude 요금제 종류, 판매 중인 상품 유무, SNS 계정과 팔로워 수]
- 하루에 쓸 수 있는 시간: [예: 평일 1시간, 주말 3시간]
- 4주 뒤 목표: [예: 첫 디지털 상품 1개 등록]

조건:

1. 주차별로 읽을 장(01~28장)과 그 주에 만들 결과물 1개씩
2. 하루 단위 할 일은 30분 안에 끝나는 크기로
3. 돈·비밀번호·개인정보가 걸린 작업은 "사람이 직접"이라고 표시
4. 표 정리하고, 마지막에 "이번 주에 절대 하지 말 것" 3가지

PART 1

Claude Code 실전 — 기초 체력

안전하게 말기고, 정확하게 확인하고, 막히면 되살린다



CHAPTER 01

Claude Code가 바꾸는 것: 챗봇과 에이전트의 차이

이 장의 결과물 — 내 업무를 "Claude Code에 맡길 것 / 챗봇에 물을 것 / 내가 직접 할 것" 셋으로 나눈 분류표 1장

챗봇은 답하고, 에이전트는 일한다

챗봇에게 사이트를 만들어 달라고 하면 코드가 화면에 나옵니다. 그걸 복사해 파일에 붙이고, 실행하고, 오류가 나면 오류 문구를 다시 복사해 붙입니다. 이 왕복에서 **읽고, 실행하고, 확인하는 사람은 모두 나**입니다. 비개발자가 바이브코딩에서 지치는 이유입니다.

Claude Code는 에이전트입니다. 공식 문서의 표현대로 "코드베이스를 읽고, 파일을 고치고, 명령을 실행하는" 도구입니다. 내 컴퓨터의 프로젝트 폴더 안에서 파일을 직접 열어 보고, 고치고, 테스트를 돌리고, 실패하면 다시 고칩니다. 내 역할은 복사·붙여넣기 담당에서 **일을 지시하고 결과를 승인하는 사람**으로 바뀝니다.

비교	챗봇 (claude.ai 대화창 등)	에이전트 (Claude Code)
파일을 읽는 주체	내가 복사해서 보여 줌	Claude가 폴더에서 직접 읽음
수정하는 주체	내가 붙여 넣음	Claude가 파일을 직접 고침
실행·테스트	내가 터미널에서 실행	Claude가 명령을 실행하고 결과를 읽음
오류 대응	오류를 복사해 다시 질문	Claude가 오류를 읽고 다시 시도
위험	틀린 답을 내가 적용	틀린 행동을 Claude가 직접 실행 할 수 있음
내가 해야 할 일	코드 옮기기	범위 정하기, 권한 관리, 결과 검증

마지막 두 줄이 PART 1의 이유입니다. 손이 있어서 빠른 만큼 잘못된 명령도 빠르게 실행합니다. 그래서 03장(권한), 06장(검증), 07장(복구)이 설치법만큼 중요합니다.

■ 에이전트 루프 한 바퀴

Claude Code에 일을 시키면 대략 이 순서로 움직입니다.

1. **읽기** — 관련 파일을 찾아 읽습니다. 읽은 내용은 컨텍스트(05장)에 쌓입니다.
2. **계획** — 무엇을 어떤 순서로 바꿀지 정합니다. 계획(Plan) 모드에서는 여기서 멈추고 승인을 기다립니다.
3. **수정** — 파일을 만들거나 고칩니다.
4. **실행** — 테스트, 빌드, 점검 스크립트 같은 명령을 돌립니다.
5. **결과 읽기** — 출력과 오류를 읽고 성공인지 판단합니다.
6. **반복 또는 보고** — 실패면 3번으로 돌아가고, 끝났다고 판단하면 보고합니다.

사람이 끼어들 지점은 세 곳입니다. 2번에서 **방향**을 승인하고, 3~4번에서 위험한 행동을 **허락하거나 막고**, 6번에서 "끝났다"는 말 대신 **증거**를 확인합니다.

■ 어디서 쓰나: 화면별 비교

같은 Claude Code 엔진을 여러 화면에서 쓸 수 있습니다. 공식 문서는 어느 화면이든 같은 엔진에 연결되므로 저장소의 CLAUDE.md, 설정, MCP 서버가 함께 적용된다고 설명합니다.

쓰는 곳	코드가 도는 곳	필요한 것	비개발자에게 좋은 점	아쉬운 점
터미널 CLI	내 컴퓨터	설치 명령 1줄	모든 기능과 명령이 가장 먼저 들어옴. 이 책의 기준	검은 화면이 낯설
VS Code-Cursor 확장	내 컴퓨터	편집기 + 확장 설치	바뀐 부분을 색으로 보여 주는 비교 화면, 파일 옆에서 대화	확장만 깔면 터미널에서 <code>claude</code> 명령은 안 생김
JetBrains 플러그인	내 컴퓨터	IDE + CLI 별도 설치	IDE 안에서 비교 화면	비개발자는 거의 안 씀
데스크톱 앱 (Code 탭)	내 컴퓨터 또는 클라우드	앱 설치 + 유료 요금제	터미널 없이 시작, 세션 여러 개를 나란히	일부 설정은 파일로 직접 수정
웹 (claude.ai/code)·모바일 앱	Anthropic 클라우드 가상 머신	유료 요금제 + GitHub 저장소 연결	컴퓨터를 꺼도 계속 일함, 휴대폰으로 확인	내 컴퓨터의 설정·파일은 못 씀. 권한 모드는 Accept edits·Plan·Auto만

처음이라면 **데스크톱 앱이나 VS Code 확장으로 화면에 익숙해지고, 명령은 터미널 기준으로 익히기**를 권합니다. 이 책의 명령과 단축키는 터미널 기준이고, 다르게 동작하는 부분은 따로 적었습니다. 클라우드 세션은 요금제 사용량에 함께 잡히며 가상 머신 요금은 따로 없습니다(2026년 10월 공식 문서).

■ 무엇을 맡기고 무엇을 남기나

말걸지 말지는 두 질문으로 정합니다. **되돌릴 수 있는가**, 그리고 **결과를 확인할 수 있는가**.

	결과를 확인할 수 있다	확인하기 어렵다
되돌릴 수 있다	바로 말긴다 — 페이지 수정, 스크립트 작성, 테스트 추가, 문서 정리	말기되 검증 수단부터 만든다 — 디자인 개선, 문구 다듬기, 리팩터링
되돌리기 어렵다	말기되 실행 버튼은 사람이 — 배포, DB 구조 변경, 대량 메일 초안	사람이 한다 — 결제·광고비 집행, 비밀번호·인증 입력, 개인정보 대량 처리

제가 실제로 지키는 "사람이 하는 일" 목록은 이렇습니다. 운영 자동화를 하면서 직접 정한 선입니다.

- 비밀번호, 인증 문자, 2단계 인증 코드 입력
- 결제 버튼, 광고비 집행 버튼, 환불 승인
- 고객 개인정보를 대량으로 내려받거나 옮기는 일
- 계좌이체를 확인하고 멤버십을 승인하는 일 (제 두 사이트의 결제 구조)
- 운영 사이트 배포와 운영 DB 변경의 **최종 실행**

자주 하는 실수 — "에이전트니까 알아서 하겠지"라고 범위 없이 맡기는 것입니다.

1. "사이트 좋게 만들어 줘"처럼 끝이 없는 지시 → 파일 수십 개가 바뀝니다. 06장 작업 카드로 범위를 정하세요.
2. 비밀번호를 대화창에 붙여 넣기 → 기록에 남습니다. 로그인은 사람이 하고 그다음만 맡기세요.
3. "됐어요"라는 보고를 그대로 믿고 배포 → 06장의 검증 증거를 받기 전에는 끝난 게 아닙니다.

■ 실습: 15분 업무 분류표 만들기

1. 메모장에 지난 2주 동안 한 일을 생각나는 대로 15~30개 적습니다. "스레드 글 쓰기", "상세페이지 이미지 수정", "판매 내역 확인"처럼 짧게 씁니다.

2. 아래 프롬프트를 claude.ai 대화창이나 Claude Code에 붙여 넣고 목록을 넣습니다.
이 단계는 파일을 다루지 않으니 어느 쪽이든 괜찮습니다.
3. 결과표에서 "Claude Code에 맡길 것" 가운데 이번 주에 해 볼 것 1개에 동그라미를 칩니다. 06장 실습에서 이 일을 작업 카드로 바꿉니다.

> 복붙 프롬프트

[대괄호]만 바꾸기

아래는 내가 1인 사업으로 하는 일 목록이야. 각 일을 표로 분류해줘.

열: 업무 | 분류(Claude Code에 맡김 / 챗봇에 물어봄 / 사람이 직접) | 이유 | 되돌릴 수 있나(예/아니오) | 결과 확인 방법

분류 기준:

- 파일을 만들거나 고치고, 실행해서 확인할 수 있는 일 → Claude Code
- 아이디어, 문구 초안, 개념 설명처럼 파일 작업이 없는 일 → 챗봇
- 비밀번호·인증 입력, 결제·광고바·환불, 개인정보 대량 처리, 운영 배포의 최종 실행 → 사람이 직접

마지막에 "이번 주에 Claude Code로 먼저 해 볼 일" 1개를 추천하고, 그 이유를 2줄로 적어줘.

업무 목록:

[여기에 15~30개 붙여 넣기]

기대 결과: 업무마다 분류와 확인 방법이 붙은 표, 그리고 추천 1개. 보통 "사람이 직접" 칸에 3~6개가 들어갑니다.

안 될 때

증상	원인	해결
거의 전부 "Claude Code"로 분류됨	기준을 무시하고 낙관적으로 분류	"되돌릴 수 없는 일은 모두 다시 보고, 사람이 할 단계만 따로 쪼개줘"라고 추가
결과 확인 방법이 "눈으로 확인"뿐	업무 설명이 너무 짧음	업무마다 결과물(파일·화면·숫자)을 한 단어씩 붙여 다시 요청
한 업무에 사람 단계와 AI 단계가 섞여 애매함	업무가 너무 큼	"업무를 단계로 쪼개고 단계마다 분류해줘"라고 요청

■ 저자 사례: 두 사이트에서 나눈 역할

mezip.kr과 주간미제를 만들 때 역할은 이렇게 나눴습니다.

Claude가 한 일	제가 한 일
회원가입·로그인·비밀번호 재설정·관리자 화면·탈퇴 기능 코드	어떤 기능이 필요한지, 무엇을 유료로 할지 결정
자동 테스트 300개 이상 작성과 실행	테스트 결과와 실제 화면을 직접 확인하고 승인
별도 AI가 해커 역할로 보안 검토, 문제 16건 수정	검토를 맡길 시점 결정, 수정 결과 재확인
약관·안내 문구 초안	최종 문구 확정
배포 명령 준비	배포 실행 승인, 입금 확인 후 멤버십 승인

왼쪽 칸이 아무리 길어져도 오른쪽 칸은 남습니다.

> 복붙 프롬프트

[대괄호]만 바꾸기

나는 [만들려는 것: 예) 유료 뉴스레터 사이트]를 Claude Code로 만들려고 해. 시작 전에 역할 분담표를 만들어줘.

열: 단계 | Claude가 할 일 | 내가 결정하거나 승인할 일 | 내가 직접 해야 하는 일(돈·비밀번호·개인정보·운영 배포)

단계: 기획 → 화면 → 기능 → 테스트 → 보안 점검 → 배포 → 운영. 단계마다 "끝났다고 판단할 증거" 1개씩.

이 장 체크리스트

- ☐ 챗봇과 에이전트의 차이(누가 읽고, 고치고, 실행하는가)를 한 문장으로 말할 수 있다
- ☐ 내가 쓸 화면(터미널·VS Code·데스크톱 앱·웹)을 하나 정했다
- ☐ 업무 분류표를 만들고 "사람이 직접" 목록을 따로 적어 두었다
- ☐ 이번 주에 Claude Code로 해 볼 일 1개를 골랐다
- ☐ 비밀번호·결제·개인정보 작업은 AI에게 넘기지 않기로 정했다

CHAPTER 03

권한과 안전장치

이 장의 결과물 — 내 프로젝트용 `.claude/settings.json`(허용·확인·거부 규칙), 상황별 권한 모드 선택 기준, 위험 명령 목록, Git 세이프 포인트 습관

안전장치는 다섯 겹입니다

겹	무엇을 막나	무엇을 못 막나
1. 권한 모드	묻지 않고 실행할 범위 전체	모드를 너무 느슨하게 고르면 그대로 실행됨
2. 권한 규칙 (settings 파일)	특정 명령·파일을 항상 허용/ 확인/거부	규칙 모양을 비껴가는 명령, 스크립트가 간접으로 읽는 파일
3. 계획(Plan) 모드	방향이 틀린 채로 수정부터 하는 것	계획 승인 후의 실수
4. 체크포인트·되감기	Claude의 파일 편집 도구로 고친 내용	명령(Bash)으로 바뀐 파일, 외부 변경
5. Git	커밋해 둔 모든 상태	커밋하지 않은 변경, DB에 들어간 데이터

한 겹만 믿으면 그 빈틈으로 사고가 납니다. 이 장에서 다섯 겹을 한 번에 세팅합니다.

■ 권한 모드 6가지

모드 (설정값)	묻지 않고 하는 일	상태 줄 표시	이럴 때
Manual (default)	읽기만	» manual mode on	낮은 코드, 돈·로그인·데이터가 걸린 작업
acceptEdits	읽기 + 작업 폴더 안 파일 수정, mkdir·mv·cp 같은 흔한 파일 명령	» accept edits on	바뀐 내용을 나중에 git diff로 몰아서 볼 때
plan	읽기 (자동 모드가 가능하면 판정 모델이 승인한 명령도)	» plan mode on	바꾸기 전에 조사와 계획만
auto	전부, 단 별도 판정 모델이 위험 행동을 차단	» auto mode on	방향을 믿을 수 있는 긴 작업
dontAsk	미리 허용한 것만, 나머지는 자동 거부	» don't ask on	자동 실행 스크립트
bypassPermissions	전부 (확인 없음)	» bypass permissions on	격리된 컨테이너, 가상 머신 전용. 이 책에서는 쓰지 않습니다

2026년 10월 공식 문서 기준.

자동(auto) 모드에 대해 꼭 알아 둘 것: 공식 문서 기준 v2.1.283 이상에서는 터미널과 VS Code의 기본 시작 모드가 auto입니다. 판정 모델은 기본적으로 `curl | bash` 같은 다운로드 후 실행, 민감한 데이터의 외부 전송, 운영 배포와 DB 마이그레이션, 강제 푸시 (force push), 커밋하지 않은 변경을 날리는 `git reset --hard`, `git clean -fd` 같은 명령을 막습니다. 그래도 공식 문서는 분명히 경고합니다. **자동 모드는 확인 요청을 줄일 뿐 안전을 보장하지 않습니다.** 민감한 작업에서는 검토를 대신하지 못합니다.

모드는 실행 중 Shift+Tab으로 바꿉니다. auto에서 시작했다면 한 번 누를 때마다 Manual → acceptEdits → plan → (다시 auto) 순서로 돌아갑니다. 시작할 때 정하려면 `claude --permission-mode plan`처럼 씁니다.

제 기준: 돈·로그인·회원 데이터·배포가 걸린 작업은 **plan으로 시작해 계획을 본 뒤 Manual로 실행**, 디자인·문구·문서처럼 되돌리기 쉬운 작업은 **auto나 acceptEdits**, `bypassPermissions`는 **쓰지 않습니다**.

기본 모드를 고정하려면 `~/.claude/settings.json`에 `"permissions": { "defaultMode": "default" }`처럼 적습니다. 주의할 점이 있습니다. 공식 문서 기준 auto와 `bypassPermissions`는 프로젝트의 `.claude/settings.json`이나 `.claude/settings.local.json`에 적으면 **적용되지 않습니다**. 저장소에 들어 있는 파일이 내 안전장치를 마음대로 풀지 못하게 막아 둔 것입니다.

■ 계획(Plan) 모드 쓰는 법

계획 모드에서 Claude는 파일을 읽고 조사 명령을 돌리고 계획을 쓰지만, 소스 파일은 고치지 않습니다.

1. Shift+Tab을 눌러 상태 줄에 `plan mode on`이 보이게 합니다. 한 번만 계획을 받고 싶으면 프롬프트 앞에 `/plan`을 붙입니다 (예: `/plan 결제 안내 페이지 추가`).
2. 아래 프롬프트처럼 조사 범위와 계획에 담을 내용을 지정합니다.
3. 계획이 나오면 Ctrl+G로 내 편집기에서 계획을 직접 고칠 수 있습니다. 빠진 조건을 여기서 써넣습니다.
4. 승인 화면에서 고릅니다. **Yes, and use auto mode**(승인 후 자동 모드), **Yes, manually approve edits**(승인 후 수정마다 확인), **No, keep planning**(계획 계속).
5. 돈·데이터가 걸린 작업은 두 번째, **수정마다 확인**을 고릅니다.

공식 모범 사례는 **바뀔 내용을 한 문장으로 말할 수 있으면 계획을 건너뛰라**고 권합니다. 계획에도 비용이 듭니다.

> 복불 프로그램

[대괄호]만 바꾸기

/plan [하려는 일: 예) 판매 페이지에 환불 규정 섹션 추가]

계획에 반드시 넣을 것:

1. 읽어 본 파일 목록과 각 파일의 역할 한 줄
2. 바꿀 파일과 새로 만들 파일 (그 외 파일은 건드리지 않는다고 명시)
3. 단계별 순서와 각 단계의 확인 방법
4. 되돌리기 어려운 작업(배포, DB, 외부 서비스 설정)이 있으면 따로 표시
5. 내가 결정해야 할 질문이 있으면 맨 위에 먼저

settings 파일로 규칙 만들기

권한 규칙은 JSON 설정 파일에 적습니다. 파일은 네 곳에 둘 수 있습니다.

범위	파일 위치	누구에게 적용	용도
사용자	~/.claude/settings.json	내 컴퓨터의 모든 프로젝트	개인 기본값, 공통 거부 규칙
프로젝트 공유	.claude/settings.json	이 폴더에서 일하는 모두 (Git에 커밋)	프로젝트 규칙
프로젝트 개인	.claude/settings.local.json	이 프로젝트의 나만	개인 예외, 시험용
관리형	회사가 배포	조직 전체	보안 정책

같은 항목이 여러 곳에 있으면 **관리형 → 실행 시 --settings → 프로젝트 개인 → 프로젝트 공유 → 사용자** 순으로 위가 이깁니다. 단, allow-deny 같은 목록은 덮어쓰지 않고 **합쳐집니다**. 권한 확인 창에서 "Yes, and don't ask again"을 고르면 그 규칙이 .claude/settings.local.json에 자동으로 저장됩니다.

CHAPTER 15

팔리는 디지털 상품 고르기

이 장의 결과물 — 상품 후보 3개의 점수표, 수요 증거 5칸, 이번에 만들 상품 1개와 "만들지 않을 이유" 메모

15권 넘게 올려 보고 알게 된 것

먼저 제 숫자를 그대로 보여 드리겠습니다. 래피드에 15권 넘게 올렸고, 판매는 한쪽으로 크게 쏠렸습니다.

상품	판매 건수	비고
「AI가 만든 사이트, 다시 디자인합니다」	112건	결제의 거의 전부가 이 한 권
「부업의 종류와 실행과정」	13건	두 번째
나머지 13권 이상	대부분 0~3건	주제는 대부분 AI 활용·바이브코딩

"전자책 15권 판매 중"이라는 말은 그럴듯하게 들리지만, 실제로는 **한 권이 팔고 나머지는 거의 팔지 못한** 구조였습니다. 이 장은 그 차이가 어디서 왔는지 따져 보고, 다음 상품을 고를 때 쓰는 기준으로 바꾸는 장입니다.

112건과 0건은 무엇이 달랐나 (가설)

판매 건수가 적어서 원인을 하나로 확정할 수는 없습니다. 아래는 **제가 세운 가설**이고, 검증된 법칙이 아닙니다. 그래도 가설을 적어 두면 다음 상품에서 확인할 수 있습니다.

비교 축	112건 상품	0~3건 상품 다수	가설
제목이 말하는 것	"다시 디자인합니다" — 결과 하나	"AI 활용법", "총정리"류의 범위	결과가 하나로 떨어지는 제목이 클릭을 부른다
대상	이미 AI로 사이트를 만들어 본 사람	AI에 관심 있는 사람 전반	문제를 이미 겪은 사람이 지갑을 연다
문제의 보이는 정도	"AI가 만든 티 나는 디자인"은 화면으로 바로 보인다	"생산성", "부업 아이디어"는 눈에 안 보인다	전후가 눈에 보이는 문제가 설득이 쉽다
같은 저자 상품끼리	겹치는 상품이 적었다	비슷한 주제가 여러 권	비슷한 책끼리 관심을 나눠 가졌을 수 있다

13건이 팔린 「부업의 종류와 실행과정」은 반대로 범위가 넓은 제목입니다. 그런데도 두 번째로 많이 팔렸습니다. 제 해석은 "부업"이라는 **검색어 자체의 수요**가 컸다는 것입니다. 즉 좁은 결과가 아니어도, 많은 사람이 이미 찾고 있는 단어 위에 올라탄 상품은 어느 정도 팔린다는 가설입니다.

확인하지 못한 변수도 적어 둡니다. 출시 시점, 그 무렵 스레드에 올린 글의 반응, 플랫폼 노출 위치, 가격은 상품마다 달랐고 저는 그것을 따로 기록하지 않았습니다. 이 책 18장의 가격 실험 기록표와 20장의 게시 기록을 만든 이유가 여기에 있습니다. **기록이 없으면 112건도 우연인지 실력인지 모릅니다.**

■ 팔릴 상품의 다섯 가지 조건

가설을 다음 상품 고르는 기준으로 바꾸면 다섯 가지가 됩니다. 후보마다 1~5점을 매깁니다.

조건	질문	5점	1점
① 문제의 선명도	구매자가 지금 겪는 장면을 한 문장으로 말할 수 있나	"배포했더니 모바일에서 버튼이 안 눌러요"	"AI를 더 잘 쓰고 싶어요"
② 결과물의 선명도	끝나면 손에 무엇이 남나	파일·화면·문서 하나	이해, 마인드, 감
③ 내가 증명 가능	내가 직접 해 본 기록이 있나	화면·숫자·실패 기록이 있다	남의 글을 정리한 것
④ 도달 가능성	그 대상이 모이는 곳에 내가 닿을 수 있나	내 채널에 이미 그 대상이 있다	어디에 있는지 모른다
⑤ 수요의 증거	이미 찾는 사람이 있다는 증거가 있나	아래 5가지 중 3개 이상	0개

합계 18점 이상이면 만들고, 13점 이하이면 보류합니다. 14~17점은 **작은 버전으로 먼저 시험**합니다. 이 기준은 제 경험에서 나온 것이니 여러분의 결과에 맞춰 조정하세요.

■ 수요를 확인하는 5가지 방법

만들기 전에 확인해야 합니다. 만들고 나서 확인하면 이미 시간이 들어간 뒤라 판단이 흐려집니다.

방법	하는 법	걸리는 시간	신호로 볼 것
1. 질문 세기	스레드·커뮤니티에서 주제 키워드로 최근 질문을 모은다	1시간	같은 고민이 서로 다른 사람에게서 반복되는가
2. 경쟁 상품 읽기	같은 플랫폼의 같은 주제 상품 5개의 가격, 구성, 화면에 보이는 판매·후기 수, 문의	1시간	팔리는 상품이 있는가, 불만이 있는가
3. 무료 글 반응	상품의 핵심 한 조각을 무료 글로 올린다	1~3일	좋아요보다 리포스트·저장 "어디서 받나요" 답글
4. 대기 명단	"완성되면 알림 받기"를 받는다 (광고성 정보 수신 동의를 분명히)	3~7일	신청자 수와 신청 이유
5. 작은 버전	체크리스트 1장·미니판을 낮은 가격이나 무료로 먼저	1~2주	실제 결제, 다운로드, 다음 상품 문의

3번은 제가 가장 크게 배운 방법입니다. 제 스레드에서 반응이 가장 컸던 글은 상품 소개가 아니라 **"배포 전 QA 프롬프트"**를 그대로 올린 글이었습니다(좋아요 476, 리포스트 304). 상품을 만들기 전에 이런 글을 먼저 던져 보면, 어떤 주제에 사람들이 "나중에 쓸 거야"라고 반응하는지 공짜로 알 수 있습니다.

4번에서 받은 연락처로 나중에 출시 소식을 보내려면 광고성 정보 수신 동의를 받아야 합니다. 동의 문구와 표시 의무는 21장에서 다룹니다.

■ 상품 형태 고르기

같은 주제라도 형태에 따라 만드는 시간과 운영 부담이 크게 다릅니다.

CHAPTER 27

AI 에이전트에게 반복 업무 맡기기

이 장의 결과물 — 내 상황에 맞는 예약 실행 방식 선택, 사람 없이 돌아가는 일을 위한 작업 명세서, 주간 리포트 예약 실행 1개, 그리고 벽에 붙여 둘 "사람이 할 일" 안전 규칙 10줄

맡길 수 있는 일, 맡기면 안 되는 일

22장 지도에서 "자동"과 "반자동" 꼬리표가 붙은 일 중 상당수는 Claude Code가 사람 없이 처리할 수 있습니다. 실제로 맡기기 좋은 일은 네 종류입니다.

유형	예	결과물
리포트	주간 판매 리포트, 콘텐츠 성과 요약	reports/ 파일
정리	글감 inbox 분류, 오래된 초안 보관 폴더로 이동	폴더 상태 변화
점검	판매 페이지 링크 확인, 안내 페이지 문장과 규정 대조	문제 목록
초안	다음 주 스레드 초안, 문의 답장 초안	drafts/ 파일

공통점은 **결과물이 파일이고, 틀려도 사람이 보기 전에는 밖으로 나가지 않는다**는 것입니다. 게시, 발송, 결제, 가격 확정처럼 밖으로 나가는 일은 마지막 한 걸음을 사람이 합니다.

사람 없이 돌리는 다섯 가지 방법

비대화형 실행 플래그 자체는 12장에서 다뤘으니, 여기서는 "1인 사업 운영에 어느 것을 쓰나"만 고릅니다(공식 문서, 2026년 10월 확인).

방법	어디서 도나	컴퓨터 켜져 있어야	내 파일 접근	이럴 때
claude -p + 내가 실행	내 컴퓨터	예	예	리포트를 원할 때 한 줄로
/loop	열어 둔 세션 안	예, 세션도 열려 있어야	예	배포 끝날 때까지 몇 분마다 확인. 반복 작업은 7일 뒤 자동 만료
데스크톱 예약 작업	내 컴퓨터 (데스크톱 앱)	예, 앱 실행·꺼져 있음	예	로컬 판매 파일로 주간 리포트
루틴(Routines)	Anthropic 클라우드	아니오	아니오(저장소를 새로 받아 씀)	노트북을 덮어도 돌아야 할 일. 최소 간격 1시간
GitHub Actions	GitHub 서버	아니오	저장소 안 파일만	저장소 기준 점검·리포트, 기록을 남기고 싶을 때

몇 가지 세부 사항이 실제 운영에서 중요합니다.

- 데스크톱 예약 작업**은 컴퓨터가 잠든 사이 예정 시각이 지나면 그 회차를 건너뛸니다. 깨어났을 때 최근 7일 안에 놓친 회차가 있으면 가장 최근 것 하나만 한 번 실행합니다. 그래서 "월요일 9시 리포트"가 밤 11시에 돌 수도 있습니다. 프롬프트에 "지난주 월~일 데이터만"처럼 기간을 못 박아 두세요. 작업마다 권한 모드를 정할 수 있고, 처음에 "지금 실행"으로 한 번 돌리며 필요한 권한을 "항상 허용"해 두면 다음부터 멈추지 않습니다.
- 루틴**은 연구 미리보기 단계이며 Pro·Max·Team·Enterprise 요금제에서 쓸 수 있습니다. claude.ai/code/routines 또는 CLI의 `/schedule`로 만듭니다. 권한 확인 없이 자율로 돌고, 만들 때 연결해 둔 커넥터가 기본으로 모두 포함되며 그 커넥터의 쓰기 기능도 묻지 않고 씁니다. **필요 없는 커넥터는 반드시 빼세요.** 실행은 구독 사용량에서 차감됩니다.
- GitHub Actions**는 `anthropics/claude-code-action@v1`을 씁니다. 인증은 API 키(`ANTHROPIC_API_KEY`) 또는 구독용 토큰(`claude setup-`

token으로 만든 값을 `CLAUDE_CODE_OAUTH_TOKEN` 비밀 값으로)입니다. 예약 실행은 UTC 기준이고, 기본 브랜치에서만 돌고, 혼잡할 때 늦어질 수 있습니다.

■ 작업 명세서: 사람 없는 실행의 계약서

대화할 때는 AI가 헛갈리면 물어보지만, 예약 실행에서는 물어볼 사람이 없습니다. 그래서 프롬프트 대신 **작업 명세서**를 씁니다. 6장의 작업 카드를 무인 실행용으로 바꾼 것입니다.

```
# 작업: 주간 판매 리포트
- 입력: data/clean/sales.csv (읽기만)
- 출력: reports/YYYY-Www.md 한 개 (새 파일만 만든다, 기존 파일 수정 금지)
- 기간: 실행 시각과 무관하게 "지난주 월요일~일요일"
- 허용: 파일 읽기, reports/ 에 쓰기
- 금지: 인터넷 접속, 메일·메시지 발송, data/raw 열기, 삭제
- 완료 조건: ops/weekly-report-prompt.md 형식의 6개 섹션
- 실패 시: reports/FAILED-YYYY-MM-DD.md 에 이유 한 줄 남기고 종료
```

이 명세서를 기준으로 실행 명령의 권한도 좁힙니다. `claude -p`로 돌린다면 허용 도구를 명시하고 반복 횟수 상한을 겁니다.

```
claude -p "$(cat ops/jobs/weekly-report.md)" \
  --allowedTools "Read,Write" --max-turns 10 --output-format json >
logs/weekly.json
```

`--max-turns`는 반복 횟수를 넘으면 오류로 끝나고, `--output-format json` 결과에는 예상 비용(`total_cost_usd`) 칸이 들어 있어 기록용으로 좋습니다(공식 문서상 클라이언트 쪽 추정치라 실제 청구와 다를 수 있습니다).

> 복붙 프롬프트

[대괄호]만 바꾸기

아래 작업을 사람 없이 매주 실행하려고 해. 실행 전에 이 명세서의 빈틈을 검토해줘.

[작업 명세서 붙여넣기]

1. 실행 시각이 밀리거나(컴퓨터 잠들·지연) 두 번 실행될 때 결과가 들어지는 부분
 2. 입력 파일이 없거나 비어 있거나 형식이 바뀌었을 때의 동작이 정해져 있는지
 3. 허용 범위가 필요 이상으로 넓은 곳(쓰기 위치, 도구, 네트워크)
 4. 개인정보·비밀 값이 출력·로그에 섞일 수 있는 곳
 5. 돈, 비밀번호·인증 문자, 개인정보 대량 처리, 외부 게시·발송 중 하나라도 건드리는지. 건드리면 "사람 단계로 분리" 제안
- 고친 명세서 전체를 다시 보여줘.

■ 브라우저 자동화 에이전트, 실제로 맡겨 보니

Claude는 브라우저를 직접 조작할 수도 있습니다(Claude Code의 Chrome 연동 등). 저는 이 방식으로 래피드 상품 등록과 스레드 예약 게시를 맡겨 왔습니다. 편리하지만, 해 보니 문서에는 없는 현실의 제약이 있었습니다.