1인 사업자가 블로그 글 한 편의 조사·작성·검수·게시를 AI 에이전트 단계로 나눠 맡긴 기록

혼자 사업하면서 블로그와 SNS까지 직접 챙기는 분께 드리는 기록입니다. 글 한 편을 AI 에이전트(AI Agent) 단계로 나눠 맡기는 순서, 사람이 꼭 확인할 자리를 정하는 기준, 그리고 내 업무에 바로 옮겨 적을 수 있는 단계 나눔표를 담았어요.

AX(AI 전환, AI Transformation) 컨설팅 'AI특별시'를 혼자 운영하고, 같은 주제의 블로그·SNS 채널 '수달빌더'(the-cyclopedia.com)도 직접 꾸립니다. 블로그에는 직접 시험한 AI 활용법과 운영 기록을 올리는데, 글 한 편을 조사부터 게시까지 AI 에이전트에 단계별로 나눠 맡겨 실제로 돌리고 있어요. 아래 설명과 이미지는 그 저장소의 파일, 실제로 돌린 명령의 출력, 운영자가 정한 운영 기준 기록에서 옮겼습니다.

① 어떤 상황에서 AI를 활용했나요?

글 한 편을 일 단위로 쪼개 보니 여섯 칸이었어요

블로그 글 한 편에 들어가는 일을 순서대로 적어 보면 이렇습니다. 원본 자료를 찾아 읽고, 원고를 쓰고, 다시 읽으며 사실을 확인하고, 공개하면 안 되는 값을 골라내고, 이미지와 검색 칸을 채워 올린 뒤, 발행할지 판단하는 일이에요. 혼자 맡으면 한 사람이 이 여섯 칸을 차례로 전부 해야 하죠.

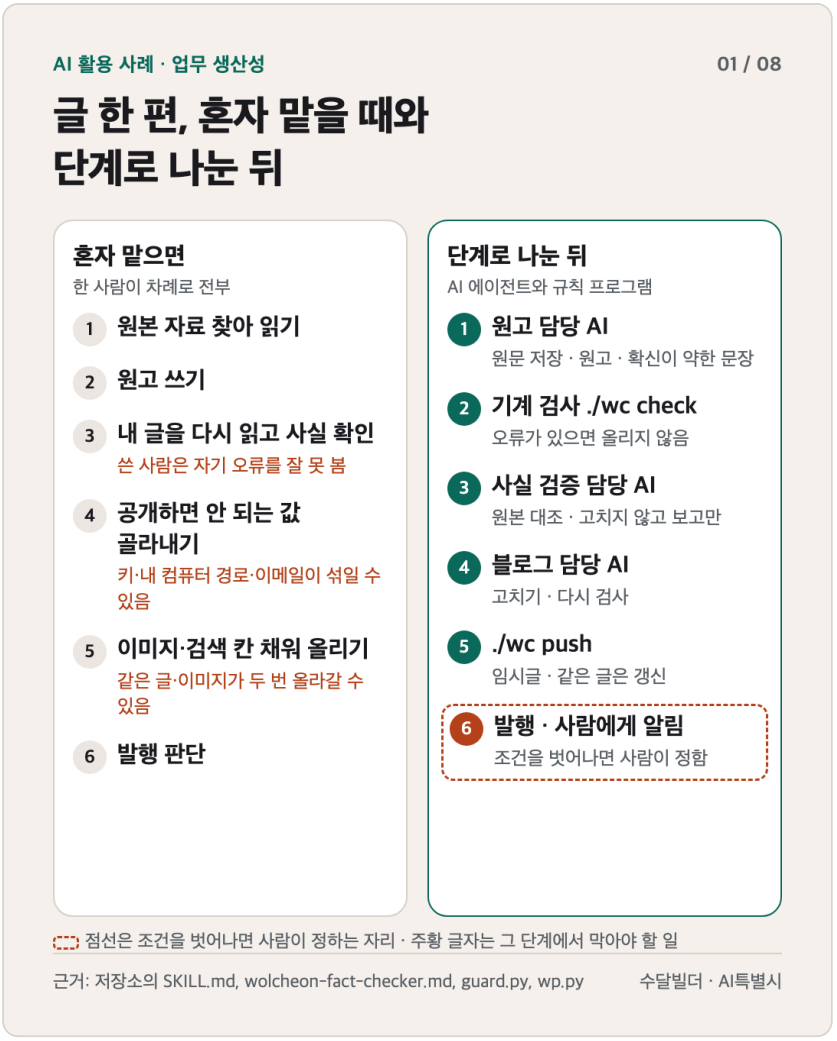
칸마다 막아야 할 일도 따로 있었습니다.

1) 사실 확인 칸: 방금 글을 쓴 사람은 자기 글의 오류를 잘 못 봅니다. 사실 검증 담당 에이전트의 정의 파일 첫머리에도 이 이유가 그대로 적혀 있어요. "글쓴이는 방금 글을 썼기 때문에 자기 글의 오류를 잘 못 본다."

2) 공개 금지 칸: 원본 자료에는 API 키, 내 컴퓨터 경로, 이메일 주소, 앱 식별 번호 같은 값이 섞여 있어서 글에 딸려 나갈 수 있습니다.

3) 올리기 칸: 같은 글을 고쳐 다시 올릴 때 글과 이미지가 두 번 올라갈 수 있어요.

이 블로그의 글은 실제 운영 기록이라서, 정의 파일의 표현대로 "원본에 없는 수치나 과장된 표현이 하나라도 나가면 블로그 전체의 신뢰가 깨진다"는 게 출발점이었습니다. 한 편에 걸리던 시간은 따로 재 둔 기록이 없어서 적지 않고, 단계와 구조로 보여 드릴게요.



이미지 1. 글 한 편, 혼자 맡을 때와 단계로 나눈 뒤

② 어떤 AI를 어떻게 활용했나요?

쓰는 AI, 대조하는 AI, 검사 프로그램을 따로 두었어요

사용한 AI 도구는 Anthropic의 Claude Code입니다. Claude Code 작업 창 하나가 '블로그 담당'을 맡고, 그 창이 필요할 때 서브에이전트(Subagent) 둘을 띄워 일을 넘겨요. 하나는 원고 담당, 하나는 사실 검증 담당입니다. 모델은 Claude Opus를 씁니다. 검사와 업로드는 AI가 아니라 직접 만든 파이썬 규칙 프로그램 ./wc가 맡는데, 명령이 16개 있어요.

절차는 Claude Code의 스킬(Skill) 파일 하나(SKILL.md)에 번호 붙은 일곱 절로 적어 두고, 문체 기준은 따로 style.md에 둡니다. 매번 같은 지시를 대화에 다시 쓰지 않고 이 파일을 읽게 하는 방식이에요.

글 한 편은 이 순서로 지나갑니다.

1) 원고 담당(서브에이전트): 공식 문서, 공식 저장소 같은 1차 출처만 근거로 쓰고, 근거가 된 원문은 파일로 저장해 둡니다. 받지 못한 페이지의 내용은 쓰지 않아요. 원고를 다 쓰면 확신이 약한 문장 5~15개를 따로 뽑아 보고 파일에 적습니다.

2) 기계 검사(./wc check): 세 가지 검사를 한 번에 돌립니다. 공개 금지 가드(guard), 검색 형식 점검(seo: 제목 길이, 메타 설명 50~110자, 이미지 대체 텍스트, 키워드 빈도 등), 문체 점검(tone: AI가 쓴 티가 나는 문장 패턴 경고)이에요.

3) 사실 검증 담당(서브에이전트): 글을 원본과 문장마다 대조해 '틀림, 근거 없음, 애매함' 셋으로 나눠 보고합니다. 글에서 계산한 합계나 비율은 직접 다시 계산하고요. 이 담당이 쓸 수 있는 도구는 읽기·찾기·명령 실행(Read, Grep, Glob, Bash)뿐이고 파일 편집 도구는 목록에 없습니다. 역할도 "글을 고치지 않는다. 틀린 곳을 찾아 보고만 한다"로 정해 뒀어요.

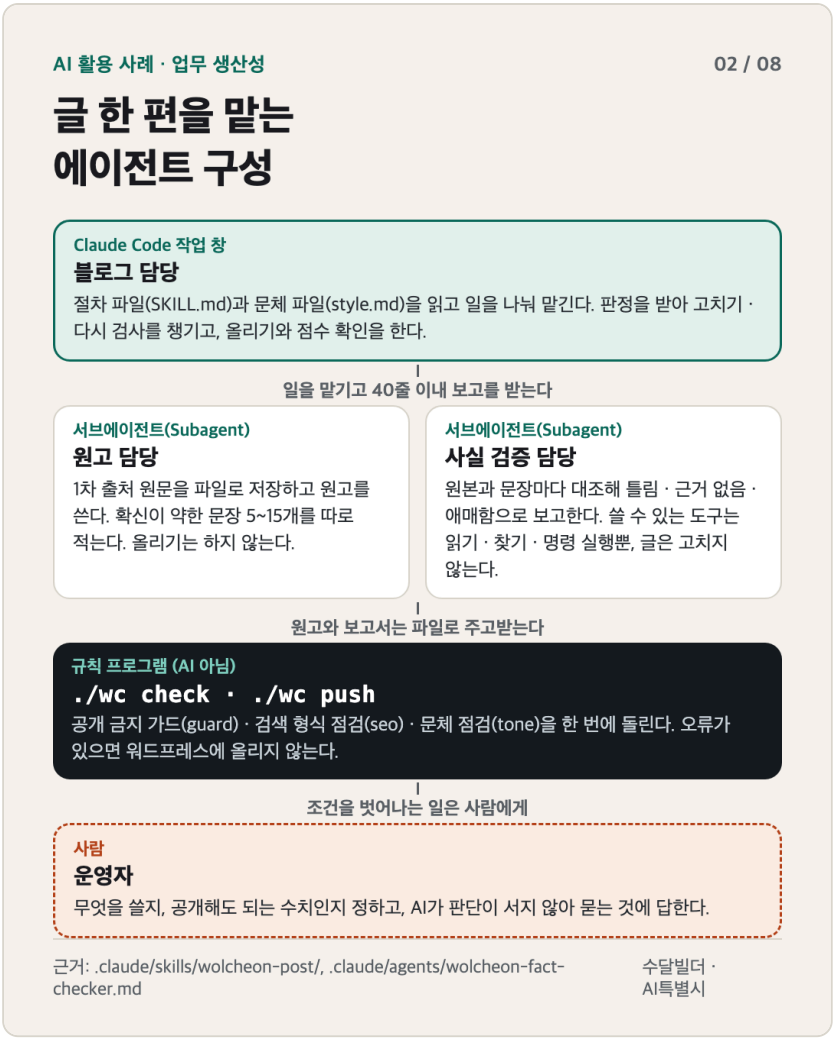
4) 블로그 담당(Claude Code 작업 창): 틀림과 근거 없음은 전부 고치거나 지우고, 고친 뒤 검사를 다시 돌립니다.

5) 올리기(./wc push): 올리기 전에 같은 검사를 먼저 돌리고, 오류가 하나라도 있으면 워드프레스에 올리지 않고 끝나요. 기본은 임시글입니다. 올린 글과 이미지의 번호는 상태 파일(state.json)에 적어 두어서, 같은 글은 새로 만들지 않고 갱신하고 같은 이미지는 다시 올리지 않습니다.

6) 점수 확인: 워드프레스 편집 화면에서 검색 노출 점검 플러그인 Rank Math의 점수가 81점 이상인지 읽고, 모자라면 감점 항목을 보고 원고를 고칩니다.

7) 발행과 알림: 조건을 다 채우면 발행하고, 운영자에게 공개한 수치 목록과 뺀 내용, 물어볼 것을 보냅니다.

블로그 담당은 담당 에이전트에게 지시를 보내고 40줄 이내의 보고를 받습니다. 원고와 근거 줄 번호 같은 긴 내용은 보고서 폴더에 파일로 남겨요.



이미지 2. 글 한 편을 맡는 에이전트 구성



이미지 3. 검사 프로그램을 실제로 돌린 화면

③ 활용 결과 어떤 변화가 있었나요?

사람 손은 세 자리에만 남았어요

혼자 맡으면 여섯 칸 모두 한 사람 몫이에요. 단계로 나눈 뒤에는 주제를 정하는 자리, 공개해도 되는지 확인하는 자리, 조건을 벗어난 발행을 판단하는 자리 세 곳만 사람이 봅니다. 나머지는 원고 담당, 검사 프로그램, 사실 검증 담당이 차례로 넘겨받습니다.

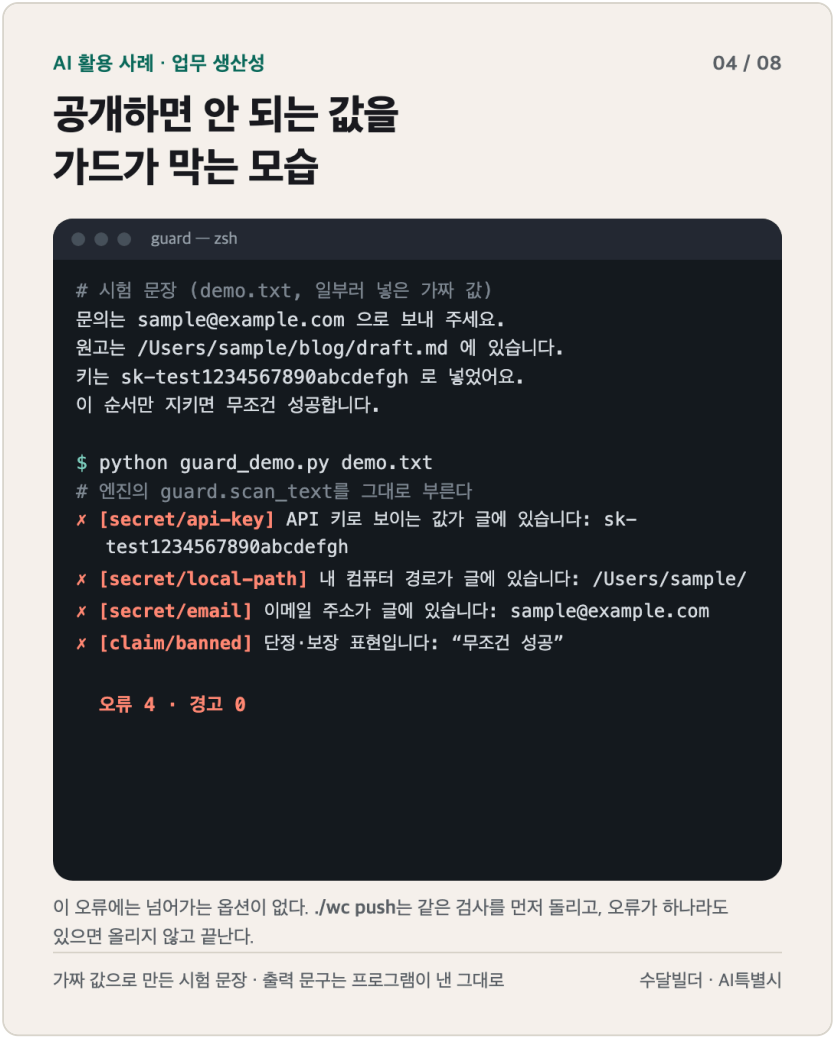
달라진 점을 기록으로 확인해 보면 이렇습니다.

1) 사실 대조가 매 편 기록으로 남습니다. 2026년 10월 20일에 보고서 폴더를 세어 보니, 사실 검증 보고서가 40건 넘게 쌓여 있었어요.

2) 공개하면 안 되는 값은 사람 눈 대신 규칙이 막습니다. API 키 모양, 내 컴퓨터 경로, 이메일 주소, 환경변수 비밀값 등 7가지 패턴과, 원본 JSON에서 이름이 id·key·token·secret·password 등으로 끝나는 키의 값, 수익 금액·보장 표현 8가지 패턴을 찾아요. 여기서 나온 오류에는 넘어가는 옵션이 없어서, 걸리면 그 값을 글에서 지워야 합니다.

3) 검사 프로그램 자체도 자동 테스트로 확인합니다. 2026년 10월 20일에 돌린 테스트 75개가 모두 통과했어요.

4) 대조 담당이 실제로 걸러 낸 문장도 남아 있습니다. 한 글의 원고는 팀 문서가 '사실 검증'이라고 부르는 일을 '사실 확인'이라고 썼고, 절차 파일에 없는 내용("사람은 판정이 갈린 곳만 봐요")을 소제목으로 썼어요. 사실 검증 담당이 두 곳을 파일 이름과 줄 번호까지 붙여 짚었고, 원고에는 고친 문장("사람은 공개할 수치와 물어볼 것만 봐요")이 들어갔습니다.



이미지 4. 공개하면 안 되는 값을 가드가 막는 모습

AI 활용 사례 · 업무 생산성 05 / 08

사실 검증 담당이 남긴 실제 보고서

 # ax-case-fact-check-before-publish 사실 검증 (2026-10-18)

판정: 수정 필요

틀림

1. 도입 2문단 “우리 팀은 이 일을 "사실 확인"이라고 불러서” — 팀 문서는 “사실 검증”으로 부른다.
`.claude/agents/wolcheon-fact-checker.md:7 · SKILL.md:78`

근거 없음

2. H2 “사람은 판정이 갈린 곳만 봐요” — SKILL.md 7절은 공개 수치 목록 · 뺀 내용 · 물어볼 것만 적음.
→ H2 “사람은 공개할 수치와 물어볼 것만 봐요”
…

맞음 확인

- 공개 금지: 경로 · 실명 · 금액 · 키 없음, ./wc check 오류 0
- 과장 문장 없음

사실 검증 담당이 실제로 남긴 보고서에서 옮겼다. 중간 줄은 줄였다(…). 이 보고서는 틀림 1건, 근거 없음 3건, 애매함 6건을 짚었고, 원고에는 고친 문장이 들어갔다.

파일: .work/reports/ax-case-fact-check-before-publish-fact.md 수달빌더 · AI특별시

이미지 5. 사실 검증 담당이 남긴 실제 보고서



이미지 6. 이 블로그에 발행된 실제 글

④ 나만의 방식 또는 개선 포인트는 무엇인가요?

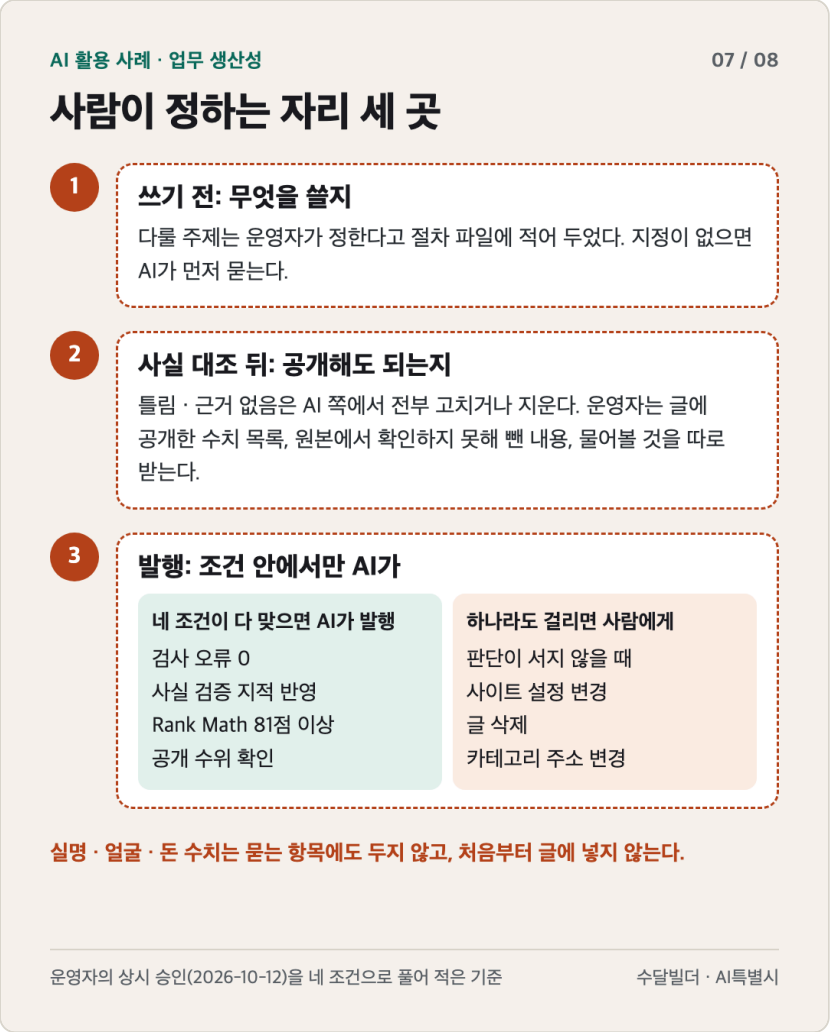
사람이 승인하는 자리를 먼저 정했어요

AI에 일을 넘기기 전에 사람이 볼 자리부터 정했습니다. 사람이 중간에 확인하는 구조를 사람 참여 방식(HITL, Human-in-the-Loop)이라고 부르는데, 이 블로그에는 그 자리가 세 곳 있어요.

1) 쓰기 전: 어떤 주제를 다룰지는 운영자가 정한다고 절차 파일에 적어 두었어요. 지정이 없으면 AI가 먼저 묻습니다.

2) 사실 대조 뒤: 틀림과 근거 없음은 AI 쪽에서 전부 고치거나 지웁니다. 운영자는 글에 공개한 수치 목록, 원본에서 확인하지 못해 뺀 내용, 물어볼 것을 따로 받아요. 절차 파일에는 이 목록을 따로 보내는 이유가 "사용자가 공개 여부를 정할 수 있게"라고 적혀 있습니다.

3) 발행: 처음에는 운영자가 "발행해"라고 말할 때만 발행했습니다. 2026년 10월 12일 운영자가 문제없으면 발행까지 하라고 상시 승인했고, 그 "문제없음"을 네 조건(검사 오류 0, 사실 검증 지적 반영, Rank Math 81점 이상, 공개 수위 확인)으로 풀어 적었어요. 네 조건이 다 맞을 때만 AI가 발행하고, 하나라도 걸리거나 판단이 서지 않으면 사람에게 묻습니다. 사이트 설정 변경, 글 삭제, 카테고리 주소 변경은 지금도 사람에게 물어요.



이미지 7. 사람이 정하는 자리 세 곳

윤리와 책임을 지키려고 둔 장치

- 원본 대조: 사실 검증 담당은 "찾지 못한 것을 맞다고 치지 않는다"는 규칙을 따릅니다. 확인하지 못한 문장은 '근거 없음'으로 올라오고, 고치거나 지우기 전에는 올리지 않아요.

- 금지 표현 가드: '누구나', '무조건', '보장', '따라만 하면' 같은 말이 수익이나 성공과 붙어 결과를 장담하는 문장, 그리고 수익·매출 금액은 검사 오류로 막습니다. 이 글에도 같은 공개 금지 가드를 돌려 오류 0을 확인했어요.

- 개인정보·실명·금액: 실명, 얼굴, 수익 금액은 글·영상·카드 어디에도 넣지 않기로 운영자가 정했습니다. 원본 자료 폴더는 읽기만 하고 고치지 않아요. 계정 정보는 저장소에 올라가지 않는 설정 파일(.env)에만 둡니다.

- 출처 표시: 근거는 1차 출처만 쓰고, 글 머리 정보에 근거 파일과 공식 문서 주소를 적어 둡니다. 원고 담당 지침에는 읽은 공식 문서 주소를 글 머리에 적고 글 끝에 '참고한 공식 문서'를 붙이라는 줄이 있고, 공식 문서 화면을 넣은 글은 캡션에 그 주소를 적었어요. 문체 점검 규칙은 한글 AI 문체 점검 도구 im-not-ai(github.com/epoko77-ai/im-not-ai)의 패턴 분류를 따랐고, 코드 첫머리에 그 분류를 따랐다고 적어 두었습니다.

- AI 생성 표시: 목소리가 든 유튜브 영상은 올릴 때 '변경·합성 콘텐츠' 표시가 기본으로 켜지도록 업로드 코드에 넣었어요. 이 출품작의 이미지 8장은 실제 화면과 HTML로 직접 그린 도식이고, AI로 생성한 그림은 없습니다.

써 보며 바꾼 점

1) 확신이 약한 문장 목록: 원고 담당이 확신이 약한 문장을 스스로 뽑아 넘기게 바꿨습니다. 문체 파일에는 이렇게 쓴 다섯 편이 검증에서 1~2건씩만 걸렸다고 적혀 있어요.

2) 발행 승인: 매번 사람이 말하던 방식에서, 운영자의 상시 승인(2026년 10월 12일)을 네 조건으로 풀어 적고 그 조건을 다 채울 때만 AI가 발행하는 방식으로 바꿨습니다.

3) 말투 규칙을 검사로: 2026년 10월 13일 운영자가 1인칭 설명 투와 이름 없이 도구를 묶어 부르는 표현을 쓰지 말라고 정했고, 이 규칙을 영상 대본과 SNS 원고 검사의 금지어 목록에 넣었어요. 블로그 글은 올리기 전 점검 목록에 같은 항목을 두었습니다.

4) 보고 길이: 담당 에이전트의 보고를 40줄 이내로 줄이고 근거는 파일로 남기게 했습니다. 규칙 파일에 적은 이유는 "담당이 돌려주는 보고가 길면 그만큼 메인이 쓸 수 있는 양이 줄어든다"예요.

⑤ 다른 사람도 따라 할 수 있나요?

작업 창 하나로 시작하는 순서

창을 여러 개 띄우거나 프로그램을 짜지 않아도 같은 구조를 가져갈 수 있어요. 이 순서를 권합니다.

1) 글 한 편(또는 보고서 한 건)을 내보낼 때 하는 일을 순서대로 한 줄씩 적습니다.

2) 그중 사람이 꼭 정할 칸에 표시합니다. 주제, 공개해도 되는 수치인지, 발행, 삭제나 설정 변경처럼 되돌리기 어려운 일부터요.

3) 칸마다 멈추는 조건을 한 줄씩 붙입니다. "원문을 못 받으면 그 내용은 쓰지 않는다", "오류가 하나라도 있으면 다음 칸으로 넘기지 않는다"처럼요.

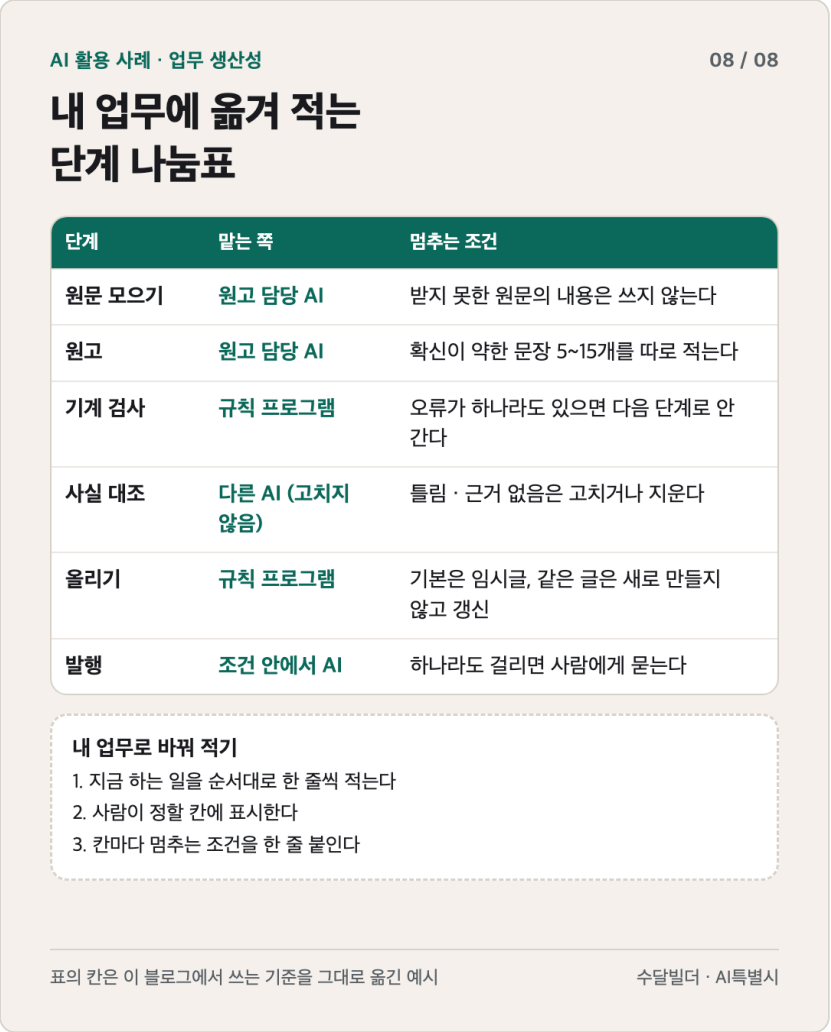
4) 기준은 파일 하나에 적어 두고 AI가 매번 그 파일을 읽게 합니다.

5) 쓰는 대화와 대조하는 대화를 나눕니다. 새 대화를 열어 글과 원문 파일만 주고 아래처럼 요청하면 됩니다.

"첨부한 글을 첨부한 원문과 문장마다 대조해 줘. 글은 고치지 마. 틀림(원문 위치를 붙여서), 근거 없음(찾아본 곳을 적어서), 애매함 셋으로만 나눠 40줄 이내로 보고해. 찾지 못한 것을 맞다고 치지 마."

6) 막을 값을 목록으로 만듭니다. 이메일, 내 컴퓨터 경로, 키 모양 문자열, 금액 표현이면 정규식 몇 줄로 찾을 수 있어요. 이 블로그의 가드 코드도 80줄 남짓입니다.

7) 올린 결과는 기록 파일에 한 줄씩 남깁니다. 같은 걸 두 번 올리는 일을 막아 줘요.



이미지 8. 내 업무에 옮겨 적는 단계 나눔표

블로그 말고도 공고문 검토, 회의록 정리, 제안서 초안처럼 "원본을 읽고 → 쓰고 → 확인하고 → 내보내는" 일이면 같은 표를 그대로 쓸 수 있습니다. 칸 이름만 바꾸면 돼요.

AI 활용 표시와 출처

- 이 출품 글의 초안과 도식은 Claude Code로 만들었고, 문장마다 저장소의 원본 파일과 대조했습니다. 운영자가 최종 내용을 확인하고 제출합니다.

- 이미지 1, 2, 7, 8: 저장소의 절차 파일(SKILL.md, style.md), 사실 검증 담당 정의 파일, 검사 코드(guard.py, wp.py, cli.py), 운영 기준 기록에 적힌 내용으로 HTML 도식을 그렸습니다.

- 이미지 3: 2026년 10월 20일 저장소에서 ./wc --help, 자동 테스트, ./wc check를 실제로 돌린 출력입니다. 일부 줄은 줄였어요.

- 이미지 4: 가드 코드(guard.scan_text)를 가짜 값으로 만든 시험 문장에 돌린 실제 출력입니다.

- 이미지 5: 사실 검증 담당이 2026년 10월 18일에 남긴 보고서에서 옮겼고, 중간 줄은 줄였습니다.

- 이미지 6: the-cyclopedia.com에 발행된 글을 2026년 10월 20일 휴대폰 화면 너비로 찍었습니다.

- 참고한 공식 문서: Claude Code 스킬 안내(code.claude.com/docs/en/skills), 서브에이전트 안내(code.claude.com/docs/en/sub-agents)