마케터 업무 자동화 매뉴얼 · 2026-10

마케터의 반복 업무를 AI 에이전트 4층으로 자동화한다

Obsidian이 기억하고, Claude Code가 쓰고, Jev가 정하고, Grok Bot이 돌린다

방법론 5단계 · 마케팅 레시피 6종 · 복사용 프롬프트 3종 · 통제 기준 · 4주 도입 로드맵

박충효 · TROE | 근거: 2026-10-25 · 10-29 웹 리서치(출처 등급 A·C 표기)

LAYER 1 · 기억	LAYER 2 · 작성	LAYER 3 · 판단	LAYER 4 · 실행
Obsidian	Claude Code	Jev	Grok Bot

CONTENTS

이 매뉴얼은 여섯 부분으로 되어 있다

개념은 짧게, 레시피와 프롬프트는 바로 따라 할 수 있게 구성했다

01
왜 지금, 무엇으로
쓰기와 고르기를 나누면 비용 구조가 바뀐다

02
자동화 5단계 방법론
한 번 해 보고, 저장하고, 그다음에 무인화

03
마케팅 업무 레시피 6종
리드·게시·경쟁사·콘텐츠·리서치·리포트

04
바로 쓰는 프롬프트 3종
볼트 세팅·skill 저장·Jev 게이트

05
통제와 리스크
사람이 승인해야 하는 선을 먼저 긋는다

06
4주 도입 로드맵
shadow로 시작해 측정한 뒤에 켠다

01

PART 1 · WHY & WHAT

왜 지금,
무엇으로 하는가

마케터 업무를 '쓰기'와 '고르기'로 나누는 것에서 시작한다

마케터의 하루는 쓰기보다 고르기가 많다

글을 만드는 일과 결정을 내리는 일을 나누면, 결정은 훨씬 싼 모델에 맡길 수 있다

쓰기 · 열린 질문

Claude Code가 맡는다

- 캠페인 카피 초안을 쓴다
- 블로그 원고를 브랜드 톤으로 고친다
- 주간 리포트 문장을 정리한다
- 랜딩 페이지 구조를 제안한다

고르기 · 닫힌 질문

Jev가 맡는다

- 이 리드에 지금 연락할까, 육성할까?
- 이 원고를 게시해도 되나?
- 이 클립은 어느 폴더로 보낼까?
- 리서치를 더 할까, 여기서 멈출까?

기준 가격 Jev는 입력 100만 토큰당 $0.042, 출력은 무료다. 예/아니오·선택형 질문만 맡긴다.

도구 네 개는 한 줄로 세우지 말고 층으로 쌓는다

네 도구를 묶는 공식 통합은 없다(2026-10-29 기준). 역할이 겹치지 않으니 층마다 하나씩 맡긴다

LAYER 1 · 기억

Obsidian 볼트

브랜드 가이드·페르소나·캠페인 기록을 Markdown으로 쌓는다

모든 에이전트가 같은 내용을 읽는다

못 하는 것
스스로 정리하지 못한다

LAYER 2 · 작성

Claude Code

원고·리포트·skill·스크립트를 로컬에서 만든다

볼트를 읽고 위키로 정리한다

못 하는 것
노트북을 끄면 멈춘다

LAYER 3 · 판단

Jev

정해 둔 선택지 중 하나를 확률·신뢰도와 함께 고른다

결정당 비용이 아주 작다

못 하는 것
글쓰기·열린 질문·계산

LAYER 4 · 실행

Grok Bot

클라우드 컴퓨터에서 브라우저·터미널로 일을 끝낸다

일정·이벤트로 routine이 돈다

못 하는 것
Bot끼리 컴퓨터를 공유한다

사람 되돌릴 수 없는 일(발행·발송·결제·삭제)은 신뢰도와 상관없이 사람이 승인한다.

무인 에이전트는 Obsidian CLI로 볼트에 붙지 못한다

공식 CLI는 데스크톱 앱이 켜져 있어야 한다. 클라우드에서 도는 Bot은 다른 길로 볼트에 닿는다

경로	조건	마케터에게 맞는 경우
파일 직접 읽기·쓰기	볼트 폴더가 내 PC에 있다	Claude Code로 원고·리포트를 만들 때. 가장 단순하다
Obsidian CLI	1.12.7+ 설치본, 앱 실행 중	검색·데일리 노트를 터미널에서 부를 때. 무인 서버엔 안 맞는다
git 저장소	볼트를 GitHub에 올린다	Grok Bot·서버 잡이 볼트를 읽을 때. 이력과 롤백이 남는다
Obsidian Headless	Node.js 22+, Sync 구독, 오픈 베타	Bot에게 컴퓨터 전체가 아닌 볼트만 열어 줄 때
MCP 서버	볼트에 닿는 MCP가 연결돼 있다	Grok Bot이 노트를 검색·저장할 때(저장소 저자 설명 기준)

권장 팀 볼트는 GitHub 저장소로 올리고, Bot은 저장소를 통해서만 읽고 쓰게 한다.

02

PART 2 · METHOD

자동화 5단계 방법론

어떤 마케팅 업무든 같은 순서로 자동화한다

한 번 해 보고, 저장하고, 그다음에 무인으로 돌린다

xAI 공식 문서의 순서(작업 → skill → routine)에 기억과 판단 단계를 끼웠다

STEP 01
쪼개기
업무를 단계로 나눈다
닫힌 질문에 표시한다

→

STEP 02
기억하기
볼트에 브랜드·고객·
과거 결정을 쌓는다

→

STEP 03
저장하기
한 번 성공한 방법을
skill로 남긴다

→

STEP 04
판단 붙이기
닫힌 질문에 Jev를 붙여
1주 동안 기록만 한다

→

STEP 05
무인화
routine으로 돌리고
승인선은 그대로 둔다

흔한 실패 3단계를 건너뛰고 바로 routine을 건다. 매시간 일정은 한 주 사용량을 하루에 쓸 수 있다.

STEP 01 · 쪼개기

업무를 단계로 쪼개고, 단계마다 담당을 정한다

예시: 매주 월요일 '경쟁사 브리프'를 6단계로 나눠 본다

단계	질문 유형	담당
1 경쟁사 5곳 사이트·광고 확인	실행	Grok Bot
2 지난주와 달라진 점 추리기	열린 질문 · 요약	Grok Bot · Claude
3 우리가 대응해야 하나?	닫힌 질문 · 예/아니오/검토	Jev
4 브리프 문장 쓰기	열린 질문 · 쓰기	Claude Code
5 팀 채널에 게시	외부 행동	사람 승인
6 볼트에 기록	되돌릴 수 있는 쓰기	자동

먼저 셀 것 닫힌 질문이 전체의 몇 %인가. 쓰기가 대부분인 업무는 Jev로 얻는 게 거의 없다.

볼트에 쌓은 맥락이 곧 에이전트의 기억이다

폴더 여섯 개면 시작할 수 있다. 에이전트는 작업 전에 이 폴더를 먼저 읽는다

brand/

브랜드 가이드

톤·금칙어·표기 규칙
원고 전에 항상 읽는다

personas/

페르소나·세그먼트

고객 유형별 고민과 언어
카피 방향을 정한다

campaigns/

캠페인 로그

날짜·예산·채널·결과
배운 점 한 줄

decisions/

결정 기록

왜 그렇게 정했나
같은 논쟁을 반복하지 않는다

competitors/

경쟁사 노트

주간 스냅샷
Bot이 매주 갱신한다

CLAUDE.md

규칙 파일

에이전트가 매번 먼저 읽는 지침
AGENTS.md로도 쓴다

STEP 03 · 저장하기

skill에는 여섯 가지를 적는다, 승인 경계가 빠지면 위험하다

xAI 공식 문서의 skill 체크리스트. 예시는 '주간 경쟁사 브리프' skill

①

언제 쓰나

매주 월요일, 경쟁사
변화 점검이 필요할 때

②

입력과 권한

경쟁사 URL 목록
광고 라이브러리 접근

③

작업 순서

사이트 → 광고 → 가격
순서로 확인

④

결과 검증

모든 주장에 링크
못 읽은 곳은 따로 표기

⑤

돌려줄 결과

달라진 점 3줄
대응 후보 최대 3개

⑥

승인이 필요한 일

고객 연락·채널 게시는
항상 승인을 받는다

Jev 질문은 선택지와 탈출구, 신뢰도 하한으로 설계한다

예시: 리드 한 건을 보고 다음 행동을 고르는 질문

Choice 질문 예시 · Python

```python
from typesafe_sdk import Choice, TypeSafeClient

with TypeSafeClient(model="jev-latest") as c:
  r = c.system_one(
    state=lead_summary,
    questions={"next": Choice(
      instructions="What to do with this lead",
      criteria={
        "contact_now": "Clear buying signal",
        "nurture": "Fit but no timing signal",
        "disqualify": "Outside target segment",
        "ask_human": "Unclear or key account",
      })})
```

선택지는 이미 있는 것에서

볼트의 폴더·세그먼트·단계 이름을 그대로 쓴다

탈출구를 넣는다

'어디에도 안 맞음'이나 ask_human을 꼭 둔다

신뢰도 하한 + 1·2위 격차

같은 입력도 확률이 0.03~0.04 흔들린다

질문 하나에 하나만

복합 질문은 반쪽만 답한다

routine은 여섯 가지를 확인하고 한도 안에서 건다

Grok Bot routine은 노트북을 덮어도 클라우드에서 돈다. 그만큼 경계를 먼저 정한다

만들 때 확인할 여섯 가지

01 담당 Bot

02 일정과 시간대

03 입력 소스

04 기대 결과

05 **승인 경계**

06 **소스가 없을 때의 행동**

항목	공식 값
시작 조건	일정, Slack·GitHub 이벤트, 이메일, webhook
Bot당 routine	최대 50개, 실행 기록 20건
일정 간격	최소 5분
실행 한도	시간당 전체 300회, routine당 200회
Test 버튼	실제 작업을 수행한다

03

PART 3 · RECIPES

마케팅 업무 레시피 6종

5단계를 실제 마케팅 업무 여섯 가지에 적용한다

자주 반복하는 마케팅 업무 여섯 가지부터 시작한다

레시피마다 어떤 층을 쓰는지 다르다. 우리 팀에 가장 아픈 것 하나만 고른다

레시피 1

리드 분류

연락·육성·제외·검토 중 고른다

Grok Bot + Jev

레시피 2

게시 전 품질 게이트

브리프 위반·근거 없는 수치면 멈춘다

Claude Code + Jev

레시피 3

경쟁사 모니터링

매주 읽고 대응 후보만 올린다

Grok Bot routine

레시피 4

콘텐츠 인박스 정리

클립을 폴더·태그로 제안한다

Obsidian + Jev

레시피 5

리서치 비용 상한

더 찾을지 멈출지 Jev가 정한다

Grok Bot + Jev

레시피 6

주간 리포트·팀 일지

무인 잡이 한 일을 한 줄로 남긴다

볼트 + routine

Grok Bot이 모으고, Jev가 나누고, 연락은 사람이 한다

리드마다 같은 질문 여러 개를 Jev에 한 번에 묻는다

01
모으기
Grok Bot이 폼·이벤트
신청 리드를 정리

→

02
나누기
Jev가 리드마다
질문 6개에 답한다

→

03
분기
연락·육성·제외
애매하면 사람 검토

→

04
반영
CRM 반영과 첫 연락은
사람이 승인

20,472건
리드 3,412건 × 질문 6개

15.7초
전체 판정 시간

$0.41
리드당 약 $0.00012

자기 보고 빌더 1인의 1회 실행이다. 내 리드 100건으로 사람 판단과의 일치율부터 잰다.

CLAUDE CODE 프롬프트

통과하지 못한 원고는 exit 1로 멈춘다

복사해서 붙여 넣기

이 프로젝트에 게시 전 품질 게이트를 만들어 줘.
— 입력: drafts/ 폴더의 원고 1개와 brand/brief.md
— Jev에게 아래 세 가지를 각각 예/아니오로 묻는다
  1) 브리프의 핵심 메시지를 벗어났나
  2) 출처 없는 수치나 주장이 있나
  3) 비밀값이나 고객 실명이 들어갔나
— 하나라도 '예'이면 exit 1로 끝내고 이유를 출력한다
— Jev 호출이 실패하거나 시간이 초과되면 게시하지 말고 나에게 묻는다
— 스크립트를 먼저 보여 주고, 내가 승인하면 hook에 연결한다

TIP 점수 스크립트가 0/1로 끝나면 Claude Code hook에 그대로 물린다.

GROK BOT 요청문

매주 읽고, 대응 후보만 올리고, 아무것도 보내지 않는다

복사해서 붙여 넣기

매주 월요일 오전 8시(Asia/Seoul)에 '주간 경쟁사 브리프' skill을 실행해 줘.
— 입력: GitHub 저장소 marketing-vault의 competitors/list.md에 있는 5곳
— 결과: 이 대화에 '달라진 점 / 대응 후보(최대 3개) / 못 읽은 곳' 세 칸으로 정리
— 고객 연락, 채널 게시, 광고 변경은 하지 않는다
— 소스를 못 읽으면 지난주 자료로 채우지 말고 실패를 보고한다
— 결과 요약 한 줄을 competitors/log.md에 남긴다

TIP 만든 직후에는 돌지 않는다. Test는 실제 작업을 하니 안전한 입력으로 시험한다.

Jev는 어디로 보낼지만 정하고, 옮기는 건 사람이 수락한다

Obsidian용 Jev 도구 세 가지. 모두 글을 고치지 않고 위치와 태그만 제안한다

JEV Inbox Router

클립을 분류해 폴더로

기본은 제안 후 수락
자동 이동은 두 문턱을 넘을 때만

Jevault

기존 폴더·태그 제안

자동 이동 기능이 없다
API 키는 Obsidian secret에

Jev the Janitor

볼트 전체 점검

전송 전 로컬 마스킹
기본 dry run, 삭제 안 함

62.7%

노트 1,408개 볼트에서 한 번만 쓰인 태그 비율

$0.0147

노트 229개를 판정하는 데 든 비용

0.55

이 신뢰도 아래는 사람 검토로 넘긴다

비싼 작업 직전에 Jev에게 한 번 묻는다

Grok Bot이 브라우저를 열고, 다시 시도하고, Bot을 더 띄우기 전에 다음 행동을 고른다

행동	뜻	고르는 조건
proceed	진행한다	다음 단계가 실패할 신호가 없다
reuse_cache	있는 걸 쓴다	같은 산출물이 이미 있다
stop_retry	재시도를 멈춘다	같은 방법이 같은 방식으로 실패했다
research_capped	리서치를 끊는다	출처가 이미 충분하다
ask_human	사람에게 묻는다	발송·게시·결제·삭제가 걸렸다

재시도 3 → 0
같은 방식 반복

페이지 10 → 4
리서치 읽은 수

53.8초 → 4.1초
후보 24개 비교

1주 shadow 결정은 기록만 하고 실행은 그대로 둔다. 매일 호출 수·행동 분포·불일치를 본다.

무인 잡이 한 일은 한 줄로 남기고, 교훈은 사람이 고른다

여러 에이전트가 같은 폴더에 작업 일지를 쓰면, 다음 세션이 그걸 읽고 이어 간다

작업 일지 예시 · 형식: 시각 | 누가 | 영역 | 한 일

```
# competitors/log.md · 2026-11-02
08:10 | grok-bot | 경쟁사 | 5곳 중 4곳 확인, 1곳 차단
08:12 | jev | 경쟁사 | 대응 필요 2건 (0.81, 0.77)
09:30 | claude | 리포트 | 주간 브리프 초안 작성
10:05 | 사람 | 리포트 | 승인 후 팀 채널 게시

# lessons.md
L3 | open | 차단된 사이트는 다음 주
     수동 확인 목록으로 넘긴다
```

한 줄 기록

누가·언제·무엇을 했는지만 사실대로 남긴다

교훈은 사람이 keep

에이전트가 남긴 교훈은 사람이 승인해야 규칙이 된다

done은 증명이 아니다

'완료' 결정이 저장·전송까지 확인해 주지는 않는다

04

PART 4 · PROMPTS

바로 쓰는 프롬프트 3종

볼트 세팅 · skill 저장 · Jev 게이트

볼트 세팅

마케팅 볼트를 만들고 규칙 파일을 쓴다

복사해서 붙여 넣기

이 폴더를 마케팅 볼트로 정리해 줘.
— 폴더: brand/ · personas/ · campaigns/ · decisions/ · competitors/ · inbox/
— CLAUDE.md에 규칙을 적는다
  · 원고를 쓰기 전에 brand/tone.md를 먼저 읽는다
  · 수치에는 출처 링크를 붙인다
  · 고객 실명과 비밀값은 쓰지 않는다
— campaigns/에 날짜·예산·채널·결과·배운 점 템플릿을 만든다
— 기존 파일은 지우거나 덮어쓰지 않는다. 바꿀 내용을 먼저 목록으로 보여 준다

TIP 같은 규칙을 AGENTS.md에도 두면 Codex 같은 다른 에이전트도 같은 지침을 읽는다.

skill 저장

한 번 성공한 과정을 skill로 남긴다

복사해서 붙여 넣기

방금 우리가 한 과정을 '주간 경쟁사 브리프' skill로 저장해 줘.
— 언제 쓰는지, 필요한 입력과 접근 권한, 작업 순서를 적는다
— 결과를 검증하는 방법과 돌려줄 형식을 적는다
— 규칙: 고객 연락과 채널 게시는 항상 내 승인을 받는다
— 브라우저 10페이지 이상, 재시도, 추가 Bot이 필요하면
먼저 jev-usage-router에 다음 행동을 묻는다
— ask_human이 나오면 신뢰도와 상관없이 멈추고 나에게 묻는다

TIP 시연으로 만든 skill은 초안이다. 실패 처리와 승인 경계를 직접 보탠다.

Jev 게이트

도구 실행 앞에 allow · ask · block 게이트를 둔다

복사해서 붙여 넣기

이 프로젝트의 도구 호출 앞에 Jev 게이트를 붙여 줘.
— 선택지는 allow · ask · block 세 가지
— 발송·게시·결제·삭제가 걸리면 Jev 답과 상관없이 ask
— Fail closed. If the Jev call errors or times out,
ask me instead of running the tool.
— 모든 결정을 decisions.jsonl에 남긴다 (API 키는 기록하지 않는다)
— 첫 1주는 shadow 모드: 결정은 기록만 하고 실행은 막지 않는다

TIP Jev는 답을 고를 뿐 승인하지 못한다. 하드 규칙은 코드가 먼저 처리한다.

05

PART 5 · CONTROL

통제와 리스크

도입 전에 사람이 승인해야 하는 선부터 긋는다

자동화는 아래 세 칸까지, 위 두 칸은 사람이 승인한다

칸이 올라갈수록 되돌리기 어렵다. 마케팅 예시로 칸마다 선을 정한다

01
읽기
경쟁사 사이트·광고
리포트 데이터 조회

→

02
준비
카피 초안·브리프
리드 분류 제안

→

03
기록·저장
볼트 기록·태그·초안 저장
언제든 되돌릴 수 있다

→

04
외부 행동
채널 게시·메일 발송
CRM 상태 변경

→

05
돈·비가역
광고 예산 변경·결제
데이터 삭제

원칙 Jev는 점수를 매길 수 있어도 승인하지 못한다. 4·5칸은 사람이 누른다.

도입 전에 팀과 합의할 리스크 여섯 가지

기술 문제보다 데이터가 어디로 나가는지, 누가 승인하는지가 먼저다

리스크	무슨 일이 생기나	대응
데이터 반출	노트 본문 최대 6,000자, 메시지 최대 4,000자가 외부 API로 나간다	고객사 폴더는 제외하고, 로컬 마스킹을 먼저 한다
공유 클라우드 컴퓨터	Grok Bot의 모든 Bot이 같은 파일·로그인을 본다	민감 자료는 올리지 않는다. 고객별 계정 분리를 검토한다
확신 있는 오답	잘못 짠 선택지는 오류 없이 높은 신뢰도로 틀린다	하한 이상도 일부를 큰 모델로 다시 검사한다
사용량 폭주	넓은 트리거 하나가 한 주 사용량을 하루에 쓴다	채널·문구를 좁히고 일정은 하루 단위부터
거짓 절감	싼 판정만 고르면 실패 전까지 효율적으로 보인다	완료된 작업당 비용으로 잰다
이름 혼동	Grok Bot(업무 Bot)과 Grok Build(코딩 에이전트)는 다르다	자료·교육에서 둘을 구분해 적는다

06

PART 6 · ROADMAP

4주 도입 로드맵

shadow로 시작하고, 측정한 뒤에 켠다

업무 하나를 4주 동안 기록 → 저장 → shadow → 무인화로 옮긴다

매주 끝나는 산출물이 있어야 다음 주로 넘어간다

1주차

쪼개기·기억

업무 1개를 단계로 분해

닫힌 질문 비율을 센다

볼트 6폴더 + CLAUDE.md

산출물: 분해표, 볼트

2주차

저장·작성

사람이 3번 성공시킨다

skill로 저장한다

게시 전 게이트를 붙인다

산출물: skill 1개

3주차

판단 shadow

Jev는 기록만 한다

매일 일치율·ask_human 확인

어긋난 질문은 선택지부터 수정

산출물: 결정 로그 7일치

4주차

무인화

routine 1개를 켠다

킬 스위치 2개 유지

첫 주는 매일 결과 확인

산출물: 첫 무인 실행 결과

호출당이 아니라 완료된 작업당 비용으로 잰다

싼 판정기는 실패하기 전까지 늘 효율적으로 보인다. 네 가지 지표를 같이 본다

지표 1

완료된 작업당 비용

모델·Bot 사용량을 끝까지
끝난 작업 수로 나눈다

지표 2

사람 판단과의 일치율

shadow 기간에 같은 건을
사람과 Jev가 각각 판정

지표 3

ask_human 비율

너무 높으면 선택지가 부족하고
0이면 하한이 너무 낮다

지표 4

재시도·중단 횟수

stop_retry가 성공했을 시도를
끊지 않았는지 확인

WRAP-UP

오늘 기억할 세 가지

01 **쓰기와 고르기를 나눈다**

글은 Claude Code, 닫힌 질문은 Jev, 실행은 Grok Bot과 코드가 맡는다

02 **한 번 해 보고, 저장하고, 그다음에 무인화한다**

skill과 1주 shadow를 건너뛰면 routine이 사용량과 신뢰를 함께 태운다

03 **되돌릴 수 없는 일은 사람이 승인한다**

발행·발송·결제·삭제는 신뢰도와 상관없이 사람 몫이다

이번 주 반복 업무 하나를 골라 단계로 쪼개고, 닫힌 질문에 동그라미를 친다.