

바이브 메이커 · 무료 자료

에이전트 데스크톱 설치 가이드

연동 기능 없는 데스크톱 프로그램도 클로드코드가 입력칸과 버튼 이름을 글로 읽고 대신 넣을 수 있습니다. 설치 한 줄부터 맥 권한 켜는 자리, 그대로 쓰는 요청 문장, 안 되는 프로그램 미리 알아보는 법까지 담았습니다.

설치 한 줄 · 붙이는 법 셋 · 맥 권한 · 요청 문장 원문 · 미리 알아보기 · 확인 단계 · 막힐 때

목차

한눈에	칸 이름을 글로 읽고, 넣고, 누르고, 다시 읽습니다	03
설치 한 줄	설치는 명령 한 줄입니다	04
클로드코드에 붙이기	붙이는 길은 셋, 한 마디로도 됐습니다	05
맥 권한	권한은 클로드코드를 띄운 앱에 줍니다	06
요청 문장	이번에 실제로 보낸 요청 문장	07
실제로 일어난 일	3분 동안 클로드코드가 한 일	08
미리 알아보기	맡기기 전에 한 번 읽혀 봅니다	09
확인 단계	「넣었다」는 응답은 「저장됐다」가 아닙니다	10
막힐 때	증상 → 할 일	11
주의	맡기기 전에 다섯 가지	12

한눈에

칸 이름을 글로 읽고, 넣고, 누르고, 다시 읽습니다

agent-desktop(에이전트 데스크톱)은 Vercel Labs 가 Apache-2.0 으로 공개한 오픈소스 명령줄 도구입니다 (GitHub 별 19,900개 · 2026-11-09). 클로드코드 같은 AI 에이전트가 데스크톱 프로그램을 보고 조작하게 해 줍니다. 화면을 사진으로 찍지 않고, 운영체제의 접근성 정보에서 **입력칸 · 버튼 이름 목록**을 글로 읽습니다.

README 원문

agent-desktop gives an AI agent computer use without screenshots or pixel matching. It reads the accessibility tree of a desktop app and returns it as JSON, with a ref for every element the agent can act on.

스크린샷도 픽셀 맞추기도 없이, 프로그램의 접근성 트리를 JSON 으로 읽고 누를 수 있는 요소마다 번호(ref)를 붙여 준다는 뜻입니다.

`snapshot` **읽기** — 칸 · 버튼 이름과 번호 목록 → `set-value` **넣기** — 번호로 찾은 칸에 값 넣기 → `click` **누르기** — 저장 버튼 누르기 → `snapshot` **다시 읽기** — 제대로 들어갔는지 확인

2분 13초 맥에서 클로드코드에 시연용 「주문관리」 앱(연동 기능 없음)을 맡겨 엑셀 주문 5건을 한 건씩 입력 · 저장하기까지 걸린 시간입니다. 끝나고 다시 읽어 엑셀과 대조하니 30칸이 모두 같았습니다 (2026-11-09 · 1회 실행).

준비물

컴퓨터
macOS 13 이상 · Windows 10 1809 이상 · Ubuntu 24.04 이상
이 자료의 실행은 맥에서만 했습니다. 윈도 · 리눅스는 문서 기준

Node.js
npm 으로 설치합니다
맥 · 리눅스는 Homebrew 로도 됩니다(4쪽)

클로드코드
터미널에서 claude 를 켤 수 있으면 됩니다
붙이는 법은 5쪽

맥 권한
손쉬운 사용 권한 하나
클로드코드를 띄운 앱에 줍니다(6쪽)

기준: github.com/vercel-labs/agent-desktop README · GitHub API(별 수, 2026-11-09 17:30) · 2026-11-09 맥 실행 기록. agent-desktop 은 제3자 오픈소스이며 바이브 메이커와 제휴 관계가 없습니다.

설치 한 줄

설치는 명령 한 줄입니다

터미널을 열고 아래 한 줄을 붙여 넣습니다. 맥 · 윈도 · 리눅스 모두 같은 줄이고, x64 · ARM64 용 실행 파일이 함께 들어옵니다(README).

⌨️ 설치 · npm — COPY → PASTE

```
npm install -g agent-desktop
```

🍺 맥 · 리눅스는 Homebrew 로도 됩니다 — COPY → PASTE

```
brew install agent-desktop
```

Bun · Yarn · pnpm 으로도 같은 패키지를 설치할 수 있습니다. 윈도용 Scoop 패키지는 아직 없습니다(README 기준).

깔렸는지 확인

✅ 상태 보기 · 운영체제 · 권한 · 읽을 수 있는 화면 종류 — COPY → PASTE

```
agent-desktop status
```

실제 출력 · 2026-11-09 맥 · 권한 부분만 줄임


```
{ "command": "status", "data": {
  "platform": "macos",  "version": "1.2.0",
  "permissions": {
    "accessibility":    { "state": "granted" },
    "automation":       { "state": "granted" },
    "screen_recording": { "state": "denied" } } } }
```


화면 기록이 denied 여도 됐습니다

이 출력 그대로 이 자료의 입력 · 저장 · 다시 읽기가 끝까지 됐습니다. 사진을 찍지 않으니까요. 화면 기록 권한은 `screenshot` 명령을 쓸 때만 필요합니다(6쪽).

accessibility 가 denied 면 아직 아무것도 읽지 못합니다. 6쪽에서 권한부터 켭니다. 이번 실행은 같은 npm 패키지(**agent-desktop 1.2.0**)를 작업 폴더에 받아 썼습니다. `-g` 전역 설치 줄은 README 기준입니다.

기준: README 「Install」(2026-11-09 열람) · agent-desktop status 실제 출력(1.2.0 · 맥)

클로드코드에 붙이기

붙이는 길은 셋, 한 마디로도 됐습니다

README 가 안내하는 길은 세 가지입니다(문서 기준). 이 자료의 실행은 셋 다 하지 않고, 요청에 **「agent-desktop 을 써」** 한 마디만 넣었습니다(아래 초록 상자).

ⓐ 스킬로 설치

🧩 **스킬 · README 원문** COPY → PASTE

```
npx skills add https://github.com/vercel-labs/agent-desktop --skill agent-desktop
```

ⓑ MCP 로 연결

🔌 **MCP 설정 · README 원문** COPY → PASTE

```
{ "mcpServers": { "agent-desktop": { "command": "agent-desktop", "args": ["mcp"] } } }
```

`agent-desktop mcp` 가 모든 명령을 MCP 도구로 내놓습니다(`desktop_snapshot` · `desktop_click` 처럼 명령 이름 그대로).

ⓒ CLAUDE.md 에 두 줄

📄 **README 원문 · README 는 AGENTS.md 라고 적었고, 클로드코드라면 CLAUDE.md** COPY → PASTE

```
Use `agent-desktop` to see and operate desktop apps. Before the first use, run `agent-desktop skills get desktop` and `agent-desktop skills get platform`, and follow them.
```

이번 실행에서 한 방식

agent-desktop 이 깔린 상태에서 요청 끝에 「agent-desktop 을 써」 한 마디만 넣었습니다. 클로드코드는 먼저 `agent-desktop --help` 를 열었고, 맨 위 안내를 보고 **가이드 두 개를 스스로 읽은 뒤** 일을 시작했습니다.

agent-desktop --help 맨 위 · 실제 출력

```
For AI agents, read this first:
  agent-desktop skills get desktop     Core guide: the snapshot/ref loop, verification, retries
  agent-desktop skills get platform    What differs on the operating system you run on
  agent-desktop <command> --help       Flags and arguments for one command
Skills are compiled into this binary, so they always match this exact version.
```

가이드는 프로그램 안에 들어 있어서, 설치한 판과 늘 같은 판을 읽습니다.

기준: README 「For AI agents」 「MCP」(문서 기준 · 직접 해 보지 않음) · 2026-11-09 실행 기록(--help 출력 · skills get desktop · platform 호출)

맥 권한

권한은 클로드코드를 띄운 앱에 줍니다

맥은 손쉬운 사용 권한이 없으면 아무것도 읽지 못합니다(`PERM_DENIED`). 권한은 셸이 아니라 **agent-desktop 을 띄운 앱**(터미널 · iTerm2 · Warp · VS Code)에 줍니다. 다른 터미널이나 편집기로 바꾸면 그 앱에도 따로 줘야 합니다.

🔐 **권한 안내 띄우기** COPY → PASTE

```
agent-desktop permissions --guide
```

1. **상태 보기** · `agent-desktop permissions` 로 denied 인 권한이 있는지 봅니다
2. **안내 띄우기** · `permissions --guide` 가 시스템 설정의 맞는 자리를 열고, 목록에 끌어 넣을 앱을 보여 주는 패널을 띄웁니다
   Touch ID 나 암호로 확인하면 다음 빠진 권한으로 저절로 넘어갑니다
3. **손으로 할 때** · 시스템 설정 › 개인정보 보호 및 보안 › 손쉬운 사용. 띄운 앱을 추가하고 켭니다
   agent-desktop 실행 파일이 목록에 따로 보이면 그것도 켭니다
4. **확인** · `agent-desktop permissions` 가 granted 면 끝. 앱을 다시 켤 필요는 없습니다
5. **읽어 보기** · `agent-desktop snapshot -i` 가 지금 앞에 있는 앱을 읽습니다

⚡ **맥 27 부터 목록 이름이 바뀌었습니다**

문서 원문 「*macOS 27 and later name this list Device Control and Data Access.*」 손쉬운 사용 자리가 안 보이면 이 이름의 목록을 찾거나, **--guide 로 바로 엽니다.**

권한	언제 필요한가	이번 실행
손쉬운 사용	항상 · 읽기 · 넣기 · 누르기 전부	granted
화면 기록	`screenshot` · `session start --screenshots` 를 쓸 때만	denied · 그래도 됨
자동화(System Events)	명령이 이 권한이 없다고 알릴 때만	granted

tmux · SSH 안에서 띄웠다면 안내 패널이 앱 대신 agent-desktop 실행 파일을 내놓습니다. 이때는 tmux 를 띄운 터미널 앱을 손으로 목록에 추가합니다(문서 기준).

기준: agent-desktop 내장 맥 가이드(skills get platform · 1.2.0) 「Requirements」 「First contact」 · 문제 해결 문서 「PERM_DENIED」 · 2026-11-09 status 출력

요청 문장

이번에 실제로 보낸 요청 문장

클로드코드에 아래 세 문장을 그대로 보냈습니다. 맡길 프로그램(「주문관리」)은 먼저 켜 두었고, 엑셀 파일은 클로드코드를 켠 폴더에 두었습니다.

💬 요청 문장 · 원문 그대로 COPY → PASTE

엑셀 파일 「11월 주문 5건.xlsx」에 있는 주문 5건을 「주문관리」 프로그램에 한 건씩 입력하고 저장해 줘. agent-desktop 을 써. 끝나면 프로그램 목록을 다시 읽어서 5건이 제대로 들어갔는지 확인해 줘.

빈칸만 바꿔 쓰는 판

✏️ 괄호 안만 바꿉니다 COPY → PASTE

엑셀 파일 「(파일 이름).xlsx」에 있는 (무엇) (N)건을 「(프로그램 이름)」 프로그램에 한 건씩 입력하고 저장해 줘. agent-desktop 을 써. 끝나면 프로그램 목록을 다시 읽어서 (N)건이 제대로 들어갔는지 확인해 줘.

위 원문에서 이름과 숫자만 비운 것입니다. 문장 구조는 같습니다.

붙이면 좋은 한 줄

+ 위 문장 뒤에 이어 붙입니다 COPY → PASTE

처음 1건을 끝까지 넣고 목록에서 확인한 다음 나머지를 넣어 줘.

이번 요청에는 이 줄이 없었습니다. 클로드코드가 스스로 **「1건을 끝까지 처리하고 목록 행 내용까지 확인한 다음 나머지 4건을 진행합니다」**라고 정하고 그 순서로 했습니다. 처음부터 적어 두면 이 순서를 직접 부탁할 수 있습니다.

보내기 전에

1. **프로그램을 먼저 켭니다**
   창이 없으면 WINDOW_NOT_FOUND 가 납니다(문서). `agent-desktop launch "앱 이름"` 으로 켤 수도 있습니다
2. **「」 안 프로그램 이름은 화면에 보이는 앱 이름으로**
   이번에 클로드코드는 `agent-desktop list-apps` 로 떠 있는 앱 목록을 보고 「주문관리」를 찾았습니다
3. **엑셀은 클로드코드가 직접 엽니다**
   이번에는 파이썬(openpyxl)으로 열어 5건 · 칸 6개를 읽었습니다

기준: 2026-11-09 맥 실행에 쓴 요청 원문 · 실행 기록(list-apps · 엑셀 읽기 · 「1건 먼저」 결정) · 내장 맥 가이드 「Identity」 · 문제 해결 문서 「WINDOW_NOT_FOUND」

실제로 일어난 일

3분 동안 클로드코드가 한 일

2026-11-09 맥에서 한 번 실행한 기록입니다. 시간은 요청을 보낸 때부터 셉니다. `--headed` 없이 돌려서 실제 마우스 · 키보드 입력은 쓰지 않았습니다. agent-desktop 의 기본이 **headless**(마우스 · 포커스를 뺏지 않음)입니다.

1	**가이드 읽기**	`--help` → `skills get desktop` · `skills get platform`	시작
2	**엑셀 · 앱 찾기**	엑셀 5건 · 칸 6개 읽기 → `status` · `list-apps` 로 「주문관리」 찾기	~25초
3	**창 읽기**	`snapshot` 한 번 · 칸 6개 + 저장 버튼이 이름과 번호로	~30초
4	**순서 정하기**	입력 명령 가이드를 더 읽고 「1건 끝까지 → 목록 확인 → 나머지」로 정함	1분 23초
5	**1번 넣고 저장**	칸 6개 `set-value` → 저장 `click` → 다시 읽어 목록 1행 확인	1분 34초
6	**2~5번 반복**	한 건마다 넣기 → 저장 → 다시 읽기. 상태 문구 「N번 주문 저장됨」	2분 13초
7	**다시 읽어 대조**	목록을 새로 읽어 엑셀과 30칸 비교 · 불일치 0 → 결과 보고	2분 58초

3번에서 처음 읽은 창 · snapshot 실제 출력(JSON)을 한 줄씩 옮김 · 칸 · 버튼 줄만

```
window "주문 관리 (시연용)"
  textfield "주문일"   @s30l9py6df7i1l:e1
  textfield "거래처"   @s30l9py6df7i1l:e2
  textfield "상품명"   @s30l9py6df7i1l:e3
  textfield "수량"     @s30l9py6df7i1l:e4
  textfield "단가"     @s30l9py6df7i1l:e5
  textfield "담당"     @s30l9py6df7i1l:e6
  button    "저장"     @s30l9py6df7i1l:e7
  statictext "저장된 주문 0건"
```

번호(ref)는 그 snapshot 안에서만 씁니다. 저장할 때마다 화면이 바뀌어서, 클로드코드는 매번 창을 새로 읽고 새 번호 (`@s1e4lv8zxrrcj6:e1` …)로 넣었습니다.

앱 = 시연용으로 만든 「주문관리」(칸 6개 + 저장 버튼 + 목록 · 연동 · 가져오기 기능 없음). 엑셀 5건도 시연용 주문 데이터입니다.

기준: 2026-11-09 맥 실행 기록(claude -p · agent-desktop 1.2.0) · 실행 기록 · 녹화 증거의 단계별 시각(요청 시작 기준으로 환산)

미리 알아보기

맡기기 전에 한 번 읽혀 봅니다

접근성 이름을 내주지 않는 프로그램은 읽을 수 없습니다. 일을 맡기기 전에 아래 한 줄로 한 번 읽혀 보면 됩니다. 앱 이름은 화면에 보이는 이름 그대로 씁니다.

🔍 **이 프로그램, 읽히나?** COPY → PASTE

```
agent-desktop snapshot --app "앱 이름" -i
```

맡길 만합니다
칸 · 버튼이 이름과 번호로 줄줄이 나온다
예: textfield "거래처" @...:e2

어렵습니다
줄이 몇 개 없거나, 이름 없는 줄만 있다
예: button null @...:e5

읽혀 본 앱 · 맥	번호 줄	이름 없음	읽힌 것 (2026-11-09)
주문관리 (시연용)	20	2	칸 6개 · 저장 버튼 · 목록 머리 7개가 이름 그대로
체스 (맥 기본 앱)	68	0	「화이트 룩, a1」처럼 칸 · 말 이름이 전부
UltiMaker Cura (3D 출력 준비)	14	4	위쪽 탭 PREPARE · PREVIEW · MONITOR 는 읽힘. 가운데 3D 작업판은 목록에 없음

게임이라고 다 안 되는 것은 아닙니다. 체스는 다 읽혔습니다. 짐작보다 **직접 읽혀 보는 쪽이 빠릅니다.**

문제 해결 문서 원문 · Empty or sparse tree

Some apps expose little accessibility data: custom-rendered UIs, games, some creative tools, and parts of web views.

화면을 직접 그리는 프로그램, 게임, 일부 창작 도구, 웹 화면 일부는 접근성 정보가 적다는 뜻입니다.

줄이 적게 나올 때 · 문서 순서

1. 권한부터 확인합니다(6쪽)
2. `-i` 를 빼고 다시 읽습니다. `--max-depth` 를 올립니다
3. `list-surfaces --app "앱 이름"` · 내용이 시트 · 팝오버 · 메뉴에 있을 수 있습니다
4. 한 번 더 읽습니다. 처음 켠 직후엔 잠깐 늦을 수 있습니다
5. `screenshot` 으로 눈 대신 봅니다(화면 기록 권한 필요)
6. 크로미움 계열 앱은 `launch "앱 이름" --cdp`

기준: 2026-11-09 snapshot 실제 출력 3개(주문관리 · 체스 · UltiMaker Cura) · 내장 맥 문제 해결 문서 「Empty or sparse tree」

확인 단계

「넣었다」는 응답은 「저장됐다」가 아닙니다

칸에 값을 넣는 `set-value` 의 응답은 **칸에 값이 들어간 것까지만** 확인합니다. 프로그램이 그 값을 받아 저장했는지는 따로 봐야 합니다.

set-value 실제 응답 · 1번 주문 「주문일」 칸 · 줄임

```
"disposition": { "delivery": "delivered_verified", "retry": "unsafe" },
"details": { "application_commit": "not_verified",
             "verification_scope": "element_value" },
"post_state": { "role": "textfield", "value": "2026-11-03" }
```

not_verified — 프로그램이 그 값을 받아들였는지는 확인되지 않았다는 뜻입니다. 그래서 **저장 버튼을 누른 뒤 창을 다시 읽어** 상태 문구와 목록 행을 봅니다.

retry: unsafe — 이미 들어갔을 수 있으니 같은 명령을 다시 보내면 **두 번 들어갈 수 있습니다.** 다시 보내도 되는 건 retry 가 safe 일 때(전달 안 됨 · not_delivered)뿐입니다. 나머지는 창을 다시 읽고 판단합니다(문서).

문서 원문 — *「Judge delivery by observation. AX return codes do not prove an effect.」* 성공 응답이 효과를 증명하지 않으니, 다시 읽어서 판단하라는 뜻입니다.

마지막 대조 · 실제 출력(목록을 새로 읽어 엑셀과 비교)

```
상태 문구: ['5번 주문 저장됨', '저장된 주문 5건']
프로그램 목록 행 수: 5     엑셀 행 수: 5
1번 | 2026-11-03 / 한빛문구 / A4 복사용지 80g / 10 / 24000 / 김지은 | 일치
2번 | 2026-11-04 / 새솔상사 / 볼펜 0.5 검정 / 40 / 800 / 박도윤 | 일치
3번 | 2026-11-05 / 다온오피스 / 스테이플러 대형 / 3 / 15500 / 이서연 | 일치
4번 | 2026-11-06 / 미래물산 / USB 메모리 64GB / 8 / 12000 / 최현우 | 일치
5번 | 2026-11-07 / 초록나라 / 사무용 의자 / 2 / 89000 / 정하늘 | 일치
불일치 행: 0 | 행 수 일치: True
```

이번 1회 실행에서 30칸이 모두 같았다는 기록입니다. 늘 틀리지 않는다는 뜻은 아니라서, **다시 읽어 확인하라는 말을 요청에 넣습니다** (7쪽 원문의 마지막 문장).

기준: 2026-11-09 실행 기록(set-value 응답 · 마지막 대조 출력, 이름 · 주문은 시연용 데이터) · 내장 가이드 「Delivery and retry」 · 내장 맥 가이드 「Judge delivery by observation」

막힐 때

증상 → 할 일

명령이 실패하면 응답 JSON 의 `error.code` 를 봅니다. 클로드코드에게 「이 오류 코드 보고 고쳐 줘」라고 해도 됩니다.

증상 · 오류 코드	할 일
`PERM_DENIED` **permissions 가 denied**	띄운 앱에 손쉬운 사용 권한을 줍니다(**6쪽**). 터미널을 바꿨으면 새 앱에도
줄이 거의 없음 · ref 없음	**9쪽** 순서대로 · 권한 → -i 빼기 → list-surfaces → 한 번 더 → screenshot → --cdp
`INVALID_ARGS` `@e8` **만 썼을 때**	번호는 snapshot 이름까지 붙여 씁니다 · `@s8f3k2p9:e8` (README)
`STALE_REF` **화면이 바뀐 뒤**	새로 snapshot 하고 새 번호로 다시 합니다
`AMBIGUOUS_TARGET` **창이 여럿**	`list-windows` 로 창 번호를 보고 `--window-id` 로 하나를 고릅니다
`WINDOW_NOT_FOUND` `APP_NOT_FOUND`	프로그램을 켜 둡니다 · `agent-desktop launch "앱 이름"`
`ACTION_FAILED`	칸이 꺼져 있거나 프로그램이 바쁩니다 · `is @ref --property enabled` 로 보고 기다립니다
`POLICY_DENIED`	마우스 끌기 · 더블클릭 · 마우스 올리기는 `--headed` 가 있어야 합니다. 실제 마우스를 써도 된다고 **허락할 때만** 붙입니다

⚡ **이번 실행에서 실제로 막힌 것 · 클로드코드 허락 설정**

요청을 비대화 실행(`claude -p`)으로 보내면서 미리 허락한 명령(agent-desktop · python3 · jq · 파일 읽기)만 쓰게 했더니, **변수 · 반복문이 섞인 명령이 거부됐습니다.** agent-desktop 오류가 아니라 클로드코드 쪽 허락 확인에서 난 일입니다.

거부 메시지 · 실제 출력

```
A variable in braces in this command can't be checked before it runs
This Bash command contains multiple operations. The following parts require approval: …
# 클로드코드가 스스로 풀어서 한 줄씩 다시 낸 명령(그대로 들어감)
agent-desktop set-value @s30l9py6df7i1l:e1 "2026-11-03"; echo; …
```

기준: README 「Quick start」 · 내장 가이드 「Refs」 · 내장 맥 문제 해결 문서 · 2026-11-09 실행 기록(거부 메시지 · 다시 낸 명령)

주의

맡기기 전에 다섯 가지

화면 내용이 출력에 남습니다

README 원문 *「Snapshots, clipboard reads, screenshots, command output, and debug artifacts can contain private desktop content. Keep them out of public logs and uploads.」* 읽은 목록 · 클립보드 · 명령 출력에 화면 속 개인 정보가 들어갈 수 있습니다. **결과 출력을 공개 글이나 단체 대화방에 그대로 올리지 않습니다.**

비밀번호는 --stdin 으로

README 원문 *「type, set-value, and clipboard-set accept --stdin for piped text, keeping it out of process arguments. Run secret steps as separate commands; batch JSON text appears in process arguments.」* 비밀번호 같은 값은 명령 줄에 쓰지 않고 따로 흘려 넣습니다.

실제 마우스는 허락할 때만

기본은 마우스 · 포커스를 뺏지 않는 headless 입니다. `--headed` 는 실제 마우스 · 키 입력을 씁니다. 내장 가이드 원문 *「Use the global --headed flag only when the user explicitly permits physical input」* · 실패를 복구하려고 headed 로 바꾸지도 않습니다(문서).

처음엔 1건만

1건 넣고, 저장하고, 다시 읽어 확인한 뒤 나머지를 맡깁니다(7쪽 「붙이면 좋은 한 줄」).

회사 프로그램은 회사 규정 먼저

자동 입력을 허용하는지, 고객 정보를 다뤄도 되는지 회사 규정부터 봅니다. 실제 업무 데이터는 처음부터 넣지 말고 연습용 데이터로 먼저 돌려 봅니다.

프로그램 끄기 단축키는 기본으로 막혀 있습니다. `cmd+q` 같은 종료 단축키는 `--force` 없이는 POLICY_DENIED 로 거절되고, 앱을 끌 때는 `close-app` 을 씁니다(내장 맥 가이드).

이 자료의 출처

agent-desktop = github.com/vercel-labs/agent-desktop (Vercel Labs · Apache-2.0 · 제3자 오픈소스). README 와 프로그램 안 가이드(1.2.0)는 2026-11-09 에 열람했고, 실행 기록은 같은 날 맥에서 한 1회 실행입니다. 윈도 · 리눅스 동작과 ⓐ ⓑ 연결은 문서 기준입니다.

더 묻고 싶으면 네이버 카페 「바이브메이커」 자료실 글 댓글로 남겨 주세요.

기준: README(2026-11-09 열람) · 내장 가이드 「Headless and headed」 · 내장 맥 가이드 「Keys」