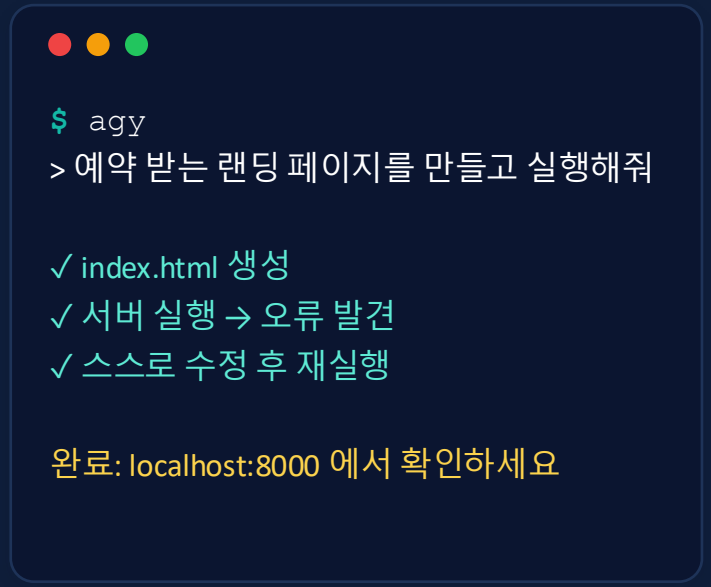
1인 창업 × 바이브 코딩

왜 터미널에서 바이브 코딩을 하는가

GUI 채팅과 CLI 에이전트를 1인 창업자의 눈으로 비교합니다



채팅 상대 vs 일하는 동료

	GUI 채팅 (웹 ChatGPT 등)	CLI 에이전트 (claude · codex · agy)
역할	답을 알려주는 조언자	**직접 일하는 실행자**
작업 범위	대화창에 붙여넣은 내용만	**프로젝트 폴더 전체**
실행과 수정	사람이 복사·붙여넣기·실행	**에이전트가 작성·실행·수정**
실수했을 때	이전 상태로 돌아가기 어려움	**git으로 언제든 되돌리기**
쌓이는 것	흩어진 대화 기록	**코드 + 규칙 파일 = 회사 자산**
서비스로 가는 길	결과물을 다시 옮겨 담아야 함	**같은 방식이 그대로 서버로**

ChatGPT의 Codex 모드도 같은 에이전트 계열입니다. 비교의 기준은 '웹이냐 터미널이냐'보다 '채팅이냐 에이전트냐'입니다.

장점 ① 복사·붙여넣기 루프에서 벗어난다

GUI 채팅: 사람이 심부름꾼

1. 질문하고 코드를 받는다
2. 코드를 복사한다
3. 파일에 붙여넣고 실행한다
4. 에러 메시지를 복사한다
5. 다시 질문한다 … 처음으로

CLI 에이전트: 사람은 감독

1. 한국어로 원하는 결과를 지시한다
2. **에이전트가 작성 → 실행 → 에러 수정 (될 때까지 스스로 반복)**
3. 결과를 확인하고 승인한다

1인 창업자의 가장 비싼 자원은 시간입니다. 반복 노동은 에이전트에게 맡기세요.

장점 ② 혼자서도 팀처럼 일한다



화면을 나눠 동시에

tmux 한 창에 에이전트 여러 명을 띄워 기획·개발·검토를 나눠 맡깁니다.

창을 닫아도 계속

연결이 끊겨도 세션이 살아 있어 긴 작업도 안심하고 맡깁니다.

한 회사에 묶이지 않음

같은 일을 세 모델에 시켜 비교하고 더 나은 결과를 고릅니다.

장점 ③ 안전하게 실험하고, 그대로 사업이 된다



되돌리기

lazygit으로 AI가 바꾼 내용을 눈으로 확인하고, 마음에 안 들면 한 번에 되돌립니다.



규칙 파일 = 노하우

CLAUDE.md·AGENTS.md에 서비스 규칙을 적어 두면 매번 설명할 필요가 없습니다.



그대로 서버로

클라우드 서버도 리눅스 터미널입니다. WSL에서 익힌 방식이 배포 환경에서 그대로 통합니다.



자동화의 부품

에이전트를 명령 한 줄로 부를 수 있어 n8n 같은 워크플로우에 노드로 끼워 넣습니다.

위 두 칸은 실패 비용을 낮추고, 아래 두 칸은 실습을 곧바로 서비스로 연결합니다.

정리: 버리는 게 아니라 용도별로 쓴다

GUI 채팅이 편할 때

- 개념 질문, 아이디어 정리
- 문서·홍보 문구 다듬기
- 짧은 코드 설명 듣기

CLI 에이전트가 필요할 때

- 실제로 돌아가는 서비스 만들기
- 여러 파일을 함께 고치는 작업
- 반복 작업 자동화와 배포

명령어는 외우지 않아도 됩니다.
터미널도 한국어로 일을 시키는 채팅창입니다. 모르면 에이전트에게 물어보세요.