2026 AI 활용 사례 공모전 · 분야 ④ 업무 생산성

AI로 회사를 지휘하다
1인 개발자의 조직형 에이전트 운영 사례

Claude Code의 에이전트 역할 분리로 1인이 PM·설계자·개발자·검수자·배포자 5역을 동시에 거느리는 가상 회사를 운영한 실사례.

분야 ④ 업무 생산성

활용 AI 도구 · Claude Code(에이전트 오케스트레이션) · Obsidian(위키) · Git worktree(개발) · Vercel(배포)
첨부 자료 · 실제 웹게임 URL(worldcupmanager.vercel.app) · GitHub 저장소 · 캡처 이미지(개인정보 노출 없음 확인)

한 줄 소개 (후보 3개)

후보 ①

Claude Code의 에이전트 역할 분리로 1인이 PM·설계자·개발자·검수자·배포자 5역을 동시에 거느리는 가상 회사를 운영한 실사례.

후보 ②

"회사 조직도를 AI에 이식한다"는 발상으로, 단일 챗봇의 한계를 넘어 목표만 주면 분해·위임·검수·통합까지 자동으로 굴리는 업무 프레임워크.

후보 ③

1인 개발자가 AI 에이전트들을 CEO 같이 지휘하는 대신, CEO 에이전트가 전문가 서브에이전트들을 조직처럼 관리하게 설계한 실운영 방식.

① 어떤 상황에서 AI를 활용했나?

1인 오너로 개발 회사를 운영할 때 근본적인 문제에 직면했습니다. 기획·설계·프론트엔드·백엔드·QA·보안·배포·문서를 전부 혼자 감당해야 했기 때문입니다. 단일 AI 챗봇에 "웹게임을 만들어줘"라고 지시하면 구현은 되지만 몇 가지 한계가 있었습니다.

첫째, 컨텍스트 뒤섞임입니다. 같은 대화창에서 설계·코드·검수·배포를 다루다 보니 누가 무엇을 담당하는지 헷갈립니다. 둘째, 검수 체계가 없습니다. 단일 AI의 산출물을 누군가 검증해야 하는데, 그 AI가 곧 만든 주체이다 보니 자기 일을 자기가 검수하는 꼴입니다. 셋째, 복잡한 작업은 완주 못 합니다. 조건과 제약이 얽혀 있으면 컨텍스트 길이 안에 다 담을 수 없었습니다.

결국 "혼자 다 할 수 없다면, AI로 회사를 차려야 한다"는 역설적 발상에 도달했습니다.

② 어떤 AI를 어떻게 활용했나?

발상의 전환: Claude Code 위에 CEO 역할을 하는 오케스트레이터 에이전트 1개 + 역할별 전문가 서브에이전트(PM·설계자·프론트엔드·백엔드·QA·보안감시·UI검수자·DevOps·테크라이터) 다수를 조직처럼 구성했습니다.

핵심 포인트: 이 사례는 완성된 게임·도구 자체가 아니라, 1인 개발자가 AI 에이전트를 다중 역할 조직으로 운영하는 방식의 케이스스터디입니다.

그림 1. CEO 오케스트레이터 + 역할별 전문가 서브에이전트 조직도

역할 분리의 핵심 아이디어:

1. **CEO 에이전트는 코드를 못 만들게 제약합니다.** CLAUDE.md라는 헌법 문서에 "CEO는 Edit/Write 도구 사용 금지"라고 명시하고, 모든 코드 변경은 빌더 에이전트에게만 위임합니다. 강제함으로써 위임이 기본 습관이 됩니다.
2. **검수는 만든 에이전트와 다른 에이전트가 합니다.** QA(기능검증)·보안감시(취약점 감사)·UI검수자(시각 일관성)는 읽기 전용 도구만 가지고 판정만 합니다. 검수자가 못 고치니까, 문제가 나면 원래 빌더에게 돌려 같은 worktree에서 고치게 합니다.
3. **인간(오너)은 최종 게이트만 봅니다.** 모든 자동 절차를 거친 후, 배포 전 오너가 직접 화면을 보고 "이거 맞아?" 판정하는 휴먼 게이트가 있습니다. AI 산출물 오류는 다중 검수 체계로 통제하고, 제품 방향은 인간의 감수성이 최종 판단합니다.

구체 절차:

목표 1줄(예: "월드컵 데이터로 감독 시뮬레이션 웹게임") → CEO가 분해(요구사항·설계·UI·코드·배포) → 병렬 위임(3개 worktree) → 각자 구현 → 검수(3라운드) → 검수자 판정 → 통합·배포 → 오너 실기기 → 최종 출시.

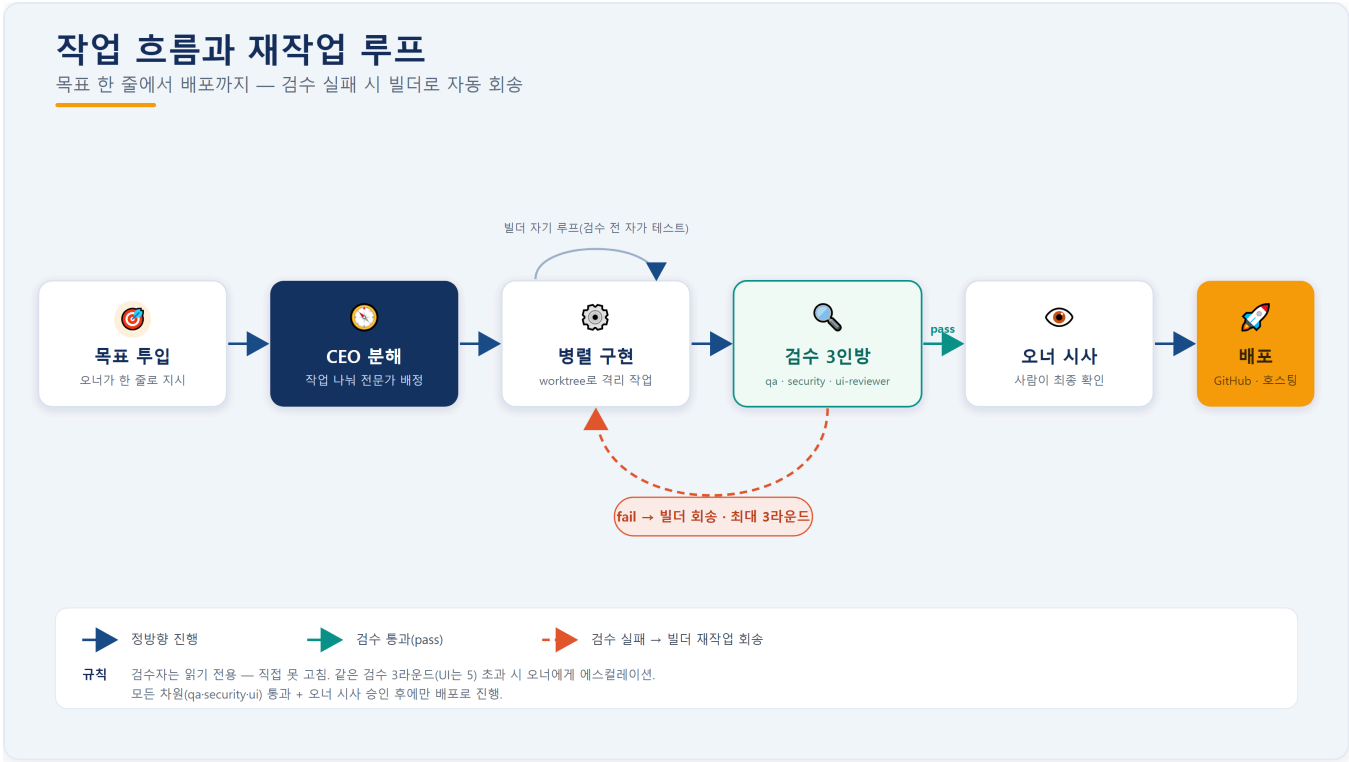


그림 2. 목표 1줄 → 분해 → 병렬 위임 → 검수 재작업 루프 → 배포 워크플로

도구·기술:

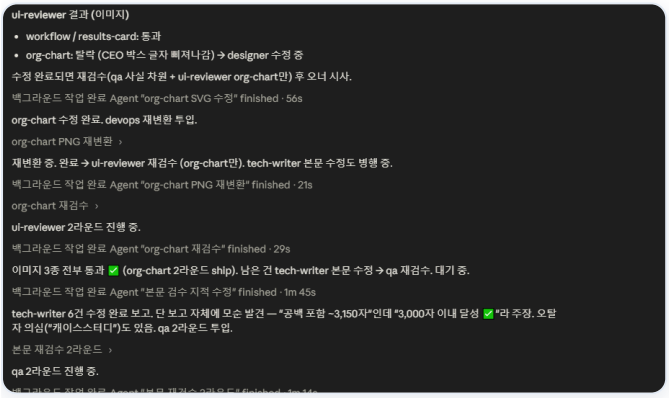
- Claude Code 에이전트 오케스트레이션 (CEO + 전문가 분할)
- Git worktree 격리 (병렬 개발, 충돌 제거)
- 역할별 .md 헌법 (토큰·책임·제약 명문화)
- 위키 영속 기억 (Obsidian, 세션 간 인수인계)
- 시크릿 분리 관리 (Infisical 환경변수)

③ 활용 결과 어떤 변화가 있었나?

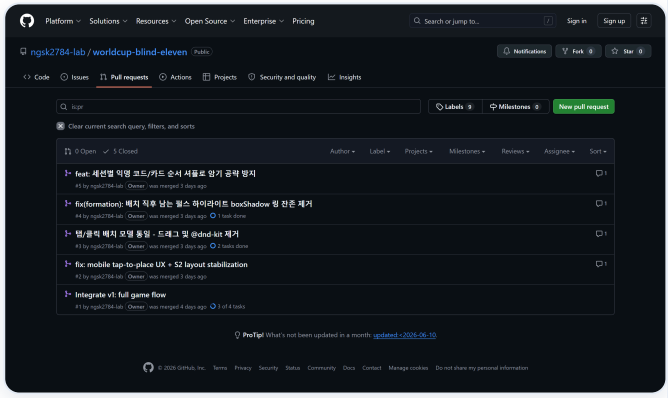
사례 1: 월드컵 웹게임 (worldcupmanager.vercel.app)

- **소요시간:** 2일 (2026-07-06~07)
- **투입 에이전트:** 3명 병렬 (데이터파이프라인·UI개발·결과연출)
- **검수:** 기능·UI 3라운드, 보안감사 1회 (npm audit 취약점 0)
- **결과:** StatsBomb 월드컵 실데이터 기반, 익명 선수 카드로 감독 경험 제공하는 인터랙티브 웹게임 완성 및 배포.
- **진행:** 내부 검수 통과 후 다커(daker.ai) 해커톤 심사 대기 중.

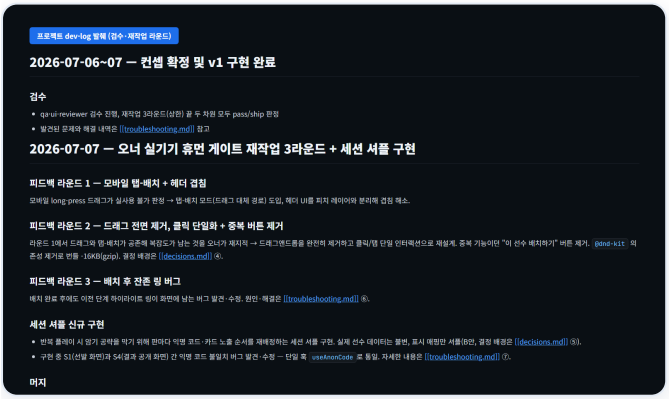
그림 3. 실운영 성과 요약 — 개발 속도 · 자동 품질관리 · 검증된 개선



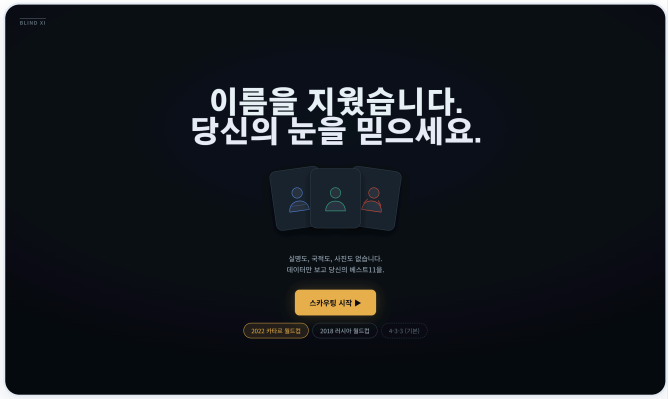
증빙 ① 오너→CEO 위임 대화



증빙 ② PR #1~#5 병렬 머지



증빙 ③ 검수·재작업 로그



증빙 ④ 배포 결과 화면

사례 2: AI 금융 계산기 (당신의 회계사, PlayMCP)

- **피드백 루프:** 외부 심사(카카오 PlayMCP) 1차 반려(서비스명 누락) → 코드 수정 → 재심사 → 통과
- **재작업:** 1라운드 (상한 3라운드 내)

- **개선:** description 5개를 일괄 수정, 원격 배포 재등록 및 검증 완료.
- **결과:** 전기요금·구독·리볼빙·멤버십·세후이자 5가지 "새는 돈"을 원 단위로 잡아내는 회계 도구 출품, PlayMCP 등록 심사 통과, 공모전 심사 결과 대기 중.

사례 3: 신약개발 제안서 (자가 레드팀 채점)

- **점수 개선:** 초판(45.5/70) → 2라운드 재작업(54.6/70) = **+9.1점**
- **개선 내역:** 독창성(SLASH 알고리즘 정식화)·도메인 사실(인용 확보·수치 교정)·성능평가(평가 방식 재정의)
- **재작업:** 2라운드 (심사위원 레드팀 시뮬레이션 → 지적사항 반영 → 재채점 → 유의미 개선 확인 후 종료)

④ 나만의 방식 또는 개선 포인트는 무엇인가?

차별점 3가지:

1. **"명령"이 아니라 "조직 위임"**: 단일 프롬프트에 모든 업무를 섞지 않고, 역할 헌법을 선언한 뒤 각 에이전트에게 독립적으로 지시합니다. 따라서 pm은 "요구사항만", backend는 "API만", qa는 "검증만" 집중할 수 있습니다. 토큰 낭비도 줄고, 컨텍스트도 명확합니다.
2. **만든 자와 검수하는 자가 다름**: AI끼리 검수 관계를 만들어 자기기만(self-deception)을 차단합니다. QA는 테스트 실행 결과만 보고 Pass/Fail 판정하며, 보안감시는 코드를 읽고 취약점만 찾습니다. 검수 불통시 문제 목록을 원래 빌더에게 돌려 같은 코드베이스에서 고치게 합니다(재작업 루프).
3. **인간은 최종 게이트만 담당**: 모든 자동 절차(위임→구현→검수→통합→배포)는 기계식으로 진행되지만, 배포 직전 오너가 한 번 실기기로 직접 써보고 "이거 맞다"고 최종 서명합니다. AI 오류는 다중 검수로 걸러지고, 제품 방향은 인간의 감수성(미적, 윤리, 비즈니스)이 결정합니다.

책임성 원칙:

- **시크릿 관리:** API 키·비밀번호는 절대 코드에 하드코딩하지 않습니다. Infisical(클라우드 볼트)에 등록하고 런타임에만 주입하며, 기록물에도 노출하지 않습니다.
- **생성형 AI 투명 고지:** 모든 산출물에 어느 부분을 AI가 만들었는지 기록합니다(이 사례 정리도 Claude를 활용했습니다).
- **한계 인식:** 토큰 비용이 크고, 장시간 작업은 컨텍스트 압축이 필요하며, 오너의 최종 판정이 없으면 제품으로 완성되지 않습니다.

⑤ 다른 사람도 따라 할 수 있나?

최소 시작 가이드 (5단계):

1. **역할 정의 .md 작성** — 당신의 팀에 필요한 역할(PM·설계자·개발자·검수자·배포자) 3~5개를 선택해, 각각 책임·제약·도구·기대 보고 형식을 한국어로 작성합니다. (특별한 코딩 지식 불필요, 텍스트만)
2. **CEO 헌법 수립** — "CEO는 코드를 직접 못 만든다", "검수자는 읽기 전용 도구만", "재작업 상한 3라운드" 같은 규칙 3~5개를 정합니다. 이를 `.claude/CLAUDE.md` 파일로 저장합니다.

3. **작은 작업부터 시작** — "블로그 글 3개 작성", "버그 1개 수정" 같은 2시간 작업으로 시작해, CEO가 분해→위임→검수→통합하는 흐름을 체험합니다.
4. **검수 게이트 추가** — 구현 완료 후 다른 역할(보안감시·UI검수)을 1개 추가해, 검수 불통 시 원래 빌더가 고치는 재작업 루프를 경험합니다.
5. **위키 기록 습관** — 작업 후마다 상태·로그·의사결정을 `.md` 파일로 기록해, 다음 세션에서 인수인계 없이 이어갈 수 있게 합니다. (이건 Obsidian 같은 도구, 또는 Github Wiki로도 가능)

독자가 기대할 수 있는 효과:

- 개인 역량 범위 내에서도 팀 같은 병렬 작업 가능
- AI 산출물 오류 자동 감지 (검수 체계)
- 복잡한 목표를 단계별로 완주 가능
- 토큰 효율 증대 (역할별 분산)
- 마지막엔 인간 판단이 최종이라 신뢰도 확보

특히 1인 프리랜서·스타트업·소규모 팀, 또는 "AI로 일을 더 빨리 하고 싶지만 품질 관리가 불안한" 업무 담당자라면 바로 적용 가능합니다. 프로그래밍 경험이 없어도, 텍스트로 역할과 규칙을 정의하기만 하면 됩니다.

생성형 AI 활용 고지

본 사례의 텍스트·콘텐츠는 생성형 AI(Claude)로 제작했으며, 구체적 활용 범위는 다음과 같습니다:

- 제출 텍스트 작성 및 구성: 초안 작성과 정리에 AI를 활용하고, 사실관계와 최종 문안은 인간이 검토·확인
- 실제 프로젝트 운영: Claude Code 에이전트 시스템 (CEO+전문가 역할 분리, 위 사례의 핵심)
- 이미지·다이어그램: SVG 코드 생성 및 최적화 (설계 아이디어는 인간)
- 데이터 수치: 실제 프로젝트 로그(Obsidian 기록, 코드 커밋, 대회 심사 결과)에서 추출
- 생성형 AI 투명성: 위 회사 헌법("CLAUDE.md")에도 동일하게 명시

본문 글자 수: 약 2,535자 (공백 제외) / 약 3,044자 (공백 포함)

제출 방식:

- 분야: ④ 업무 생산성
- 제목: "AI로 회사를 지휘하다: 1인 개발자의 조직형 에이전트 운영 사례"
- 활용 AI 도구: Claude Code(에이전트 오케스트레이션), Obsidian(위키), Git worktree(개발), Vercel(배포)
- 첨부 자료: 실제 웹게임 URL(worldcupmanager.vercel.app), GitHub 저장소 링크(공개 레벨), 캡처 이미지(개인정보 노출 없음 확인)