

ERP 개발·운영을 위한 AI 에이전트 시스템 설계

건설 시공사 ERP 백엔드를 개발하고 운영하면서, 반복되는 작업을 역할별 에이전트에 나눠 맡겼습니다. 장애 대응, 구버전 API 이관, 리뷰 반영, 컨벤션 점검, 문서 정합, 문의 분류가 대상입니다. 핵심은 자동화의 양이 아니라, 에이전트가 틀렸을 때 운영에 들어가지 않게 하는 구조입니다.

작성 하성진	역할 백엔드 · 에이전트 설계	기반 Claude Code · Codex	기준일 2026.10
-----------	---------------------	---------------------------	----------------

47

하네스 구성 요소 (연결 110)

8

실제 운영 중인 에이전트 사례

3

사람이 결정하는 지점 — 머지·배포·해결
처리

01 설계 방향

에이전트 한 명에게 모든 일을 맡기지 않았습니다. 개발 수명주기를 7단계로 나누고, 단계마다 책임과 권한이 다른 에이전트·스킬·훅을 배치했습니다.

원칙	구현
승인 경계는 코드로 고정	에이전트는 PR까지만 만든다. dev·main 직접 push와 머지는 훅이 차단한다.
판단 근거는 원천 문서	도메인 문서(Notion)와 코드 규칙(S3)을 세션마다 받아 온다. 대화 기억에 기대지 않는다.
사실은 코드가 계산	발생 횟수·시각·재발 여부는 스크립트가 API에서 계산한다. 모델은 숫자를 쓰지 않는다.
확신이 없으면 손대지 않음	자기검증에 걸린 수정안은 코드 대신 근거를 담은 이슈로 남긴다.
권한은 업무 단위로 최소화	운영 DB에 도메인별 읽기 전용 계정 8개. 운영 API는 조회만 허용한다.

02 하네스 지도

에이전트·스킬·훅·스크립트·CI가 수명주기의 어느 단계에서 무엇을 호출하는지 한 장으로 봅니다. 지도는 자산 파일에서 **자동 생성**되고, 실제 구성과 지도가 어긋나면 **push가 차단**됩니다. 문서가 실제와 달라질 수 없게 만든 장치입니다.

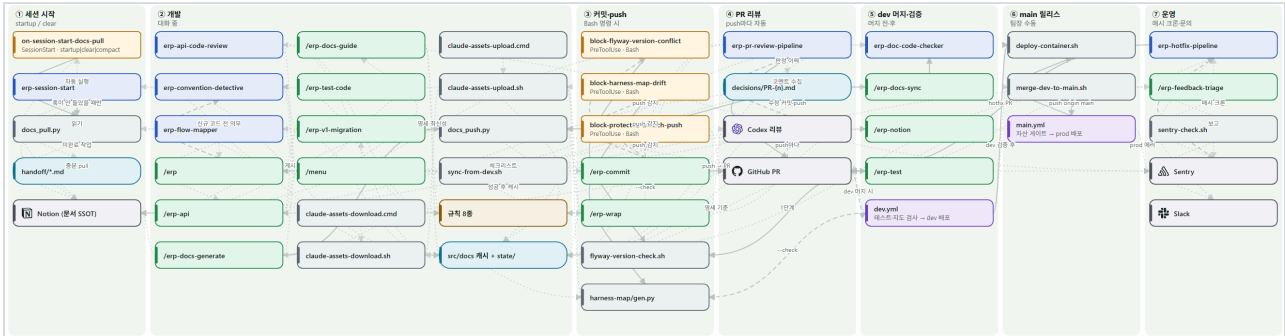


그림 1. 전체 7단계 — 노드 47 · 연결 110 (훅 4 · 에이전트 7 · 스킬 13 · 스크립트 12 · CI 2 · 규칙 1 · 데이터 3 · 외부 서비스 5)

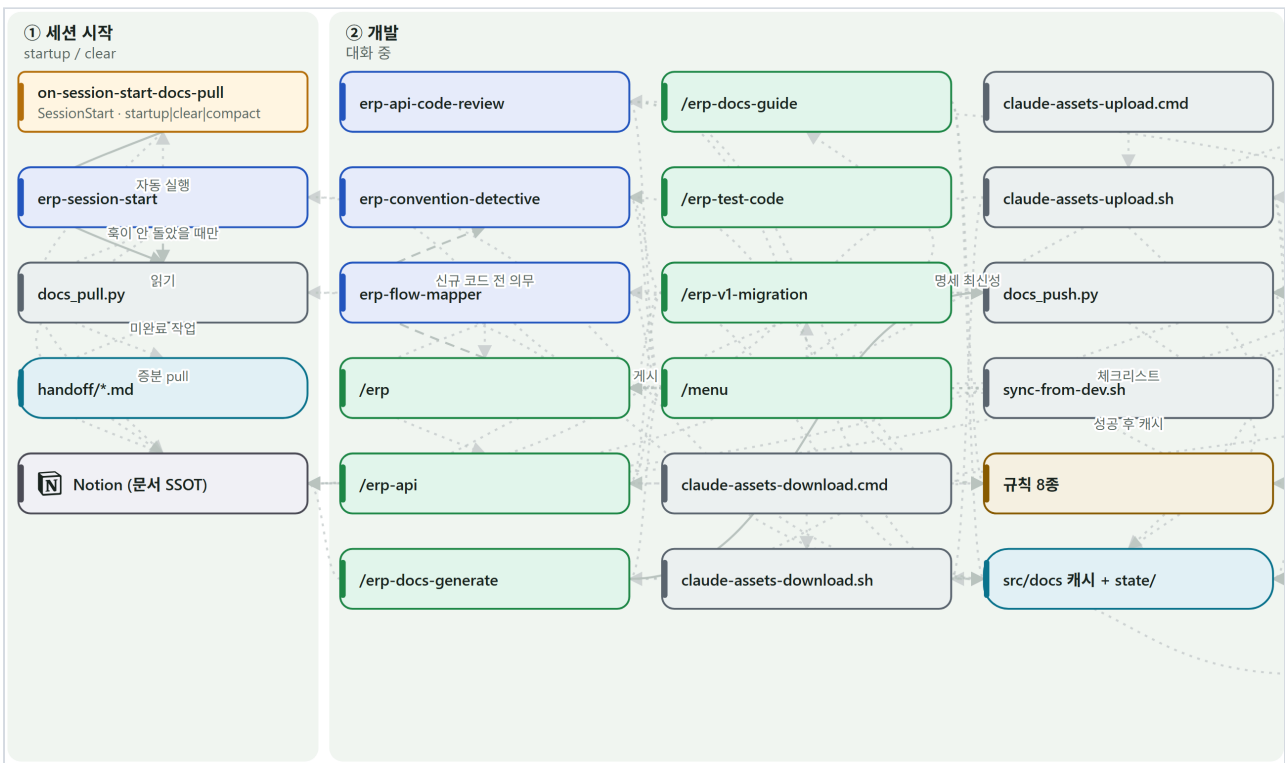


그림 2. ① 세션 시작 · ② 개발 — 세션이 열리면 Notion 문서 캐시를 갱신하고 규칙 브리핑을 주입한다. 개발 중에는 컨벤션 점검·코드 리뷰·플로우 시각화 에이전트와 V1 이관·테스트 코드·문서 생성 스킬을 쓴다.

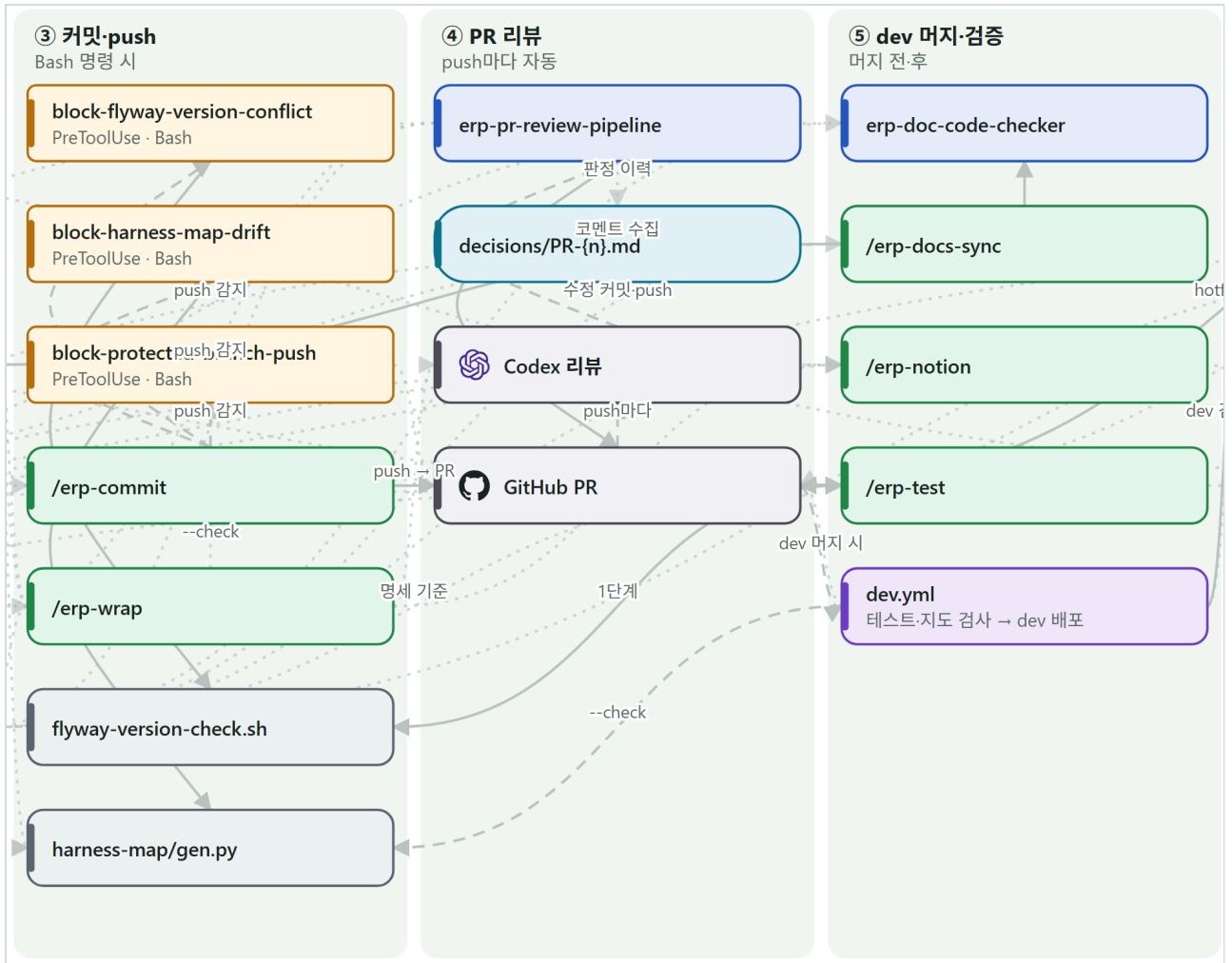


그림 3. ③ 커밋·push · ④ PR 리뷰 · ⑤ dev 머지·검증 — push 직전 혹은 3개(보호 브랜치, DB 마이그레이션 번호 충돌, 지도 불일치)가 검사한다. PR마다 Codex 리뷰가 붙고, 리뷰 반영 파이프라인이 판정 이력을 남긴다.

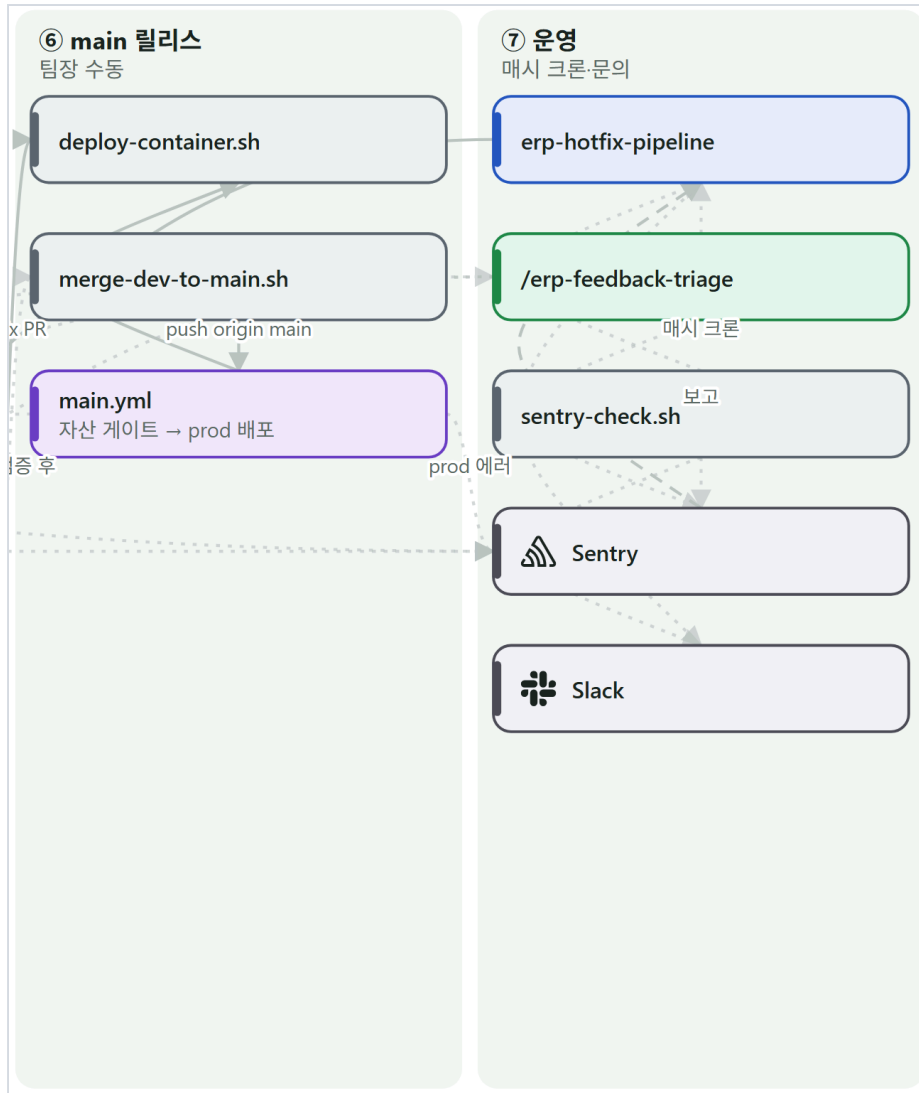


그림 4. ⑥ main 릴리스 · ⑦ 운영 — 운영 단계에서는 메시 핫픽스 파이프라인이 Sentry를 보고, 문의 트리아가 사내 문의를 분류한다.

03 에이전트 사례

각 사례는 해결한 문제 → 동작 방식 → 안전장치 → 실적 순서입니다. 실적은 저장소·GitHub·작업 기록에서 직접 센 값이며, 세는 기준을 함께 적었습니다.

3.1 운영 장애 대응 – 핫픽스 파이프라인

문제	개발자 1명이 운영하는 서버라 퇴근 후 장애가 다음 날까지 방치됐다.
동작	클라우드 예약 루틴이 업무시간 외 매시간 실행된다. 운영 Sentry 오류 조회 → 도메인 문서를 진단보다 먼저 읽음 → 운영 코드 기준 진단 → 자기검증 → dev 기준 수정 PR 또는 근거 이슈 → Slack 오류별 스레드 보고.
안전장치	자기검증 3종(과거 수정의 되돌림인가 · 근거 코드가 지금 오류를 내는가 · 도메인 문서가 금지하는가). 빌드 미확인 시 draft PR. 실행당 PR 2건·이슈 3건 상한. 운영 DB는 SQL 전달만. 재발 판정은 Sentry 기록 기준.
실적	자동 산출 12건 — 코드 PR 1 · 근거 이슈 11 (GitHub 라벨 기준). 도메인 지식 문서 9종. 파이프라인 정의 변경 커밋 19건.

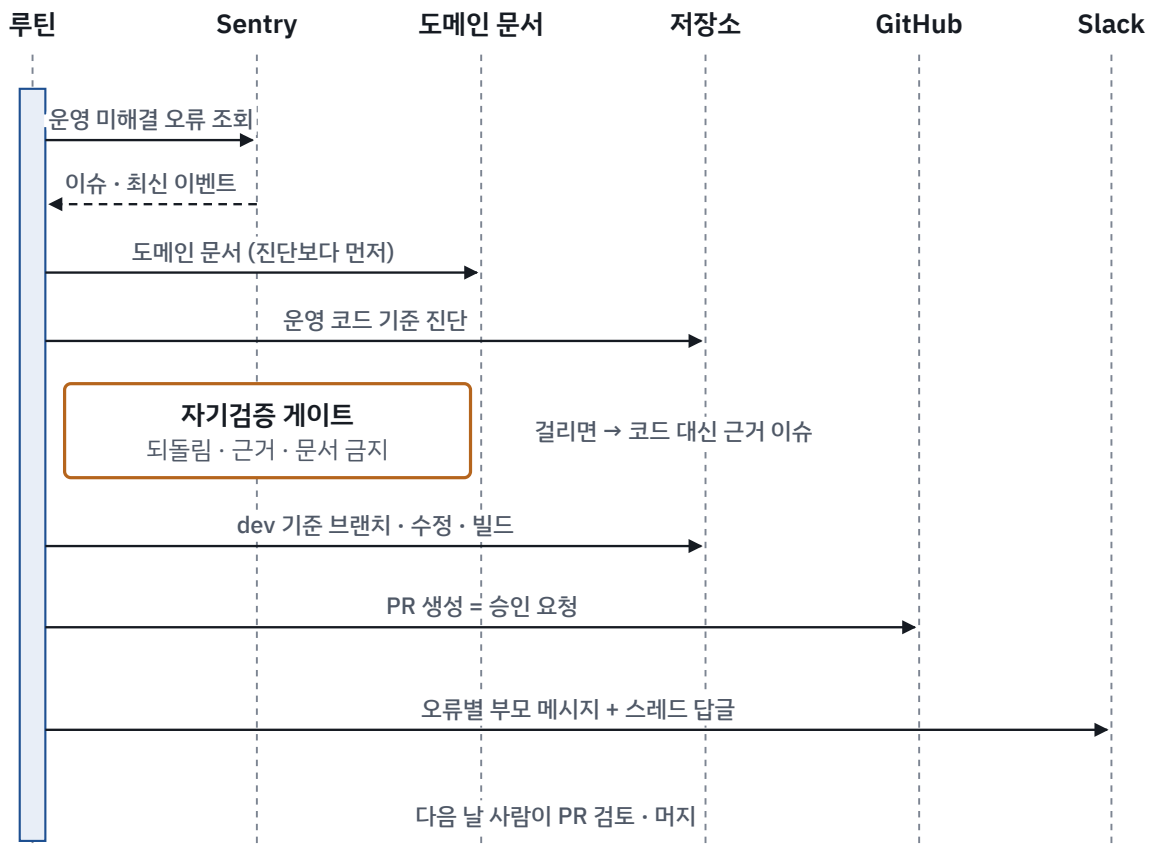


그림 5. 진단은 운영 코드, 수정은 dev 기준. 사람의 개입은 PR 머지(승인) · 배포 · Sentry 해결 처리 세 곳.

교정 사례. 시험 실행에서 나온 수정안이 3주 전 커밋의 되돌림이었고, 그 커밋은 **2,512회** 나던 오류를 막은 수정이었다. 이 사건이 자기검증 게이트의 출발점이다. 또 보고 수치가 운영 9회를 18회로 적은 일이 있어 (이슈 API가 전 환경 합계를 반환), 수치 계산을 스크립트로 분리했다.

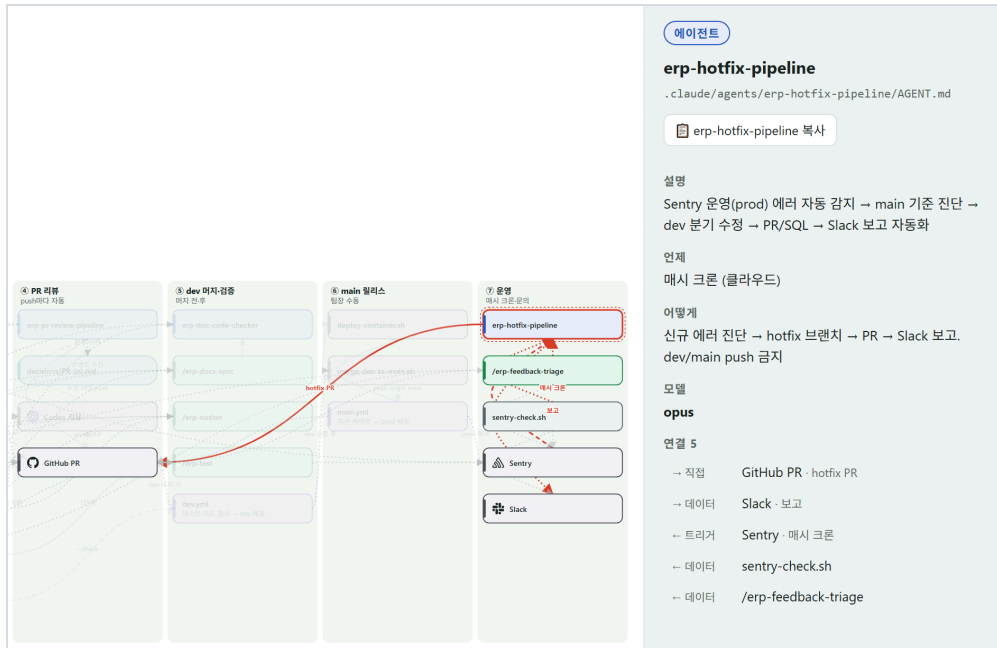


그림 6. 하네스 지도에서 핫픽스 파이프라인을 선택한 화면 — 트리거(Sentry·메시 크론), 산출(GitHub PR), 보고(Slack)가 강조된다.

3.2 구버전 API 이관 — V1 → V2 마이그레이션 파이프라인

문제 구버전(C#·Dapper·저장 프로시저)의 업무 로직을 새 서버(Spring Boot·JPA)로 옮길 때, 산식과 응답이 조금만 달라도 현장 정산이 어긋난다.

동작 ① V1 라우트 → 서비스 → 저장 프로시저 연결을 표로 정리 ② 프로시저 원문 수집 ③ 분석 문서(결과셋 순서, 조합 로직, 날짜·숫자 포맷, 채번) ④ 컨벤션 점검 에이전트 호출 후 JPA → QueryDSL 순으로 재구현 ⑤ 컴파일 → 머지 전 검증 → V1·V2 응답 비교 → 문서화.

안전장치 dev DB 조회는 3중 방어 래퍼만(단일 구문 · 첫 키워드 화이트리스트 · 읽기 전용 트랜잭션). V1 저장소는 읽기 전용. 응답 컬럼·이름·산식을 임의로 바꾸지 않음. 기준은 **운영 프로시저** — 개발 프로시저만 보고 분석했다가 되돌린 경험에서 정함.

실적 실제 API 이관 커밋 13건 (커밋 제목 분류) — 계약관리, 자재관리·거래처·자재품목, 안면인식기 근태 등록, 식당 관리, 메뉴 V2, 노무비대장(V1 응답 정합·산식 6건). V1 자료 55개 파일 · 7개 도메인.

3.3 리뷰 반복 처리 — PR 리뷰 반영 파이프라인

문제	AI 리뷰를 여러 차례 반영하다 보면 코드가 기획(스토리보드·명세)에서 조금씩 멀어진다.
동작	① 기준 문서 고정 — 바꾸면 안 되는 3종(산식·API 계약·확정 정책) 추출 ② 새 코멘트만 수집 ③ 반영 / 기각 / 에스컬레이션 판정 ④ 최소 수정 후 빌드 확인 ⑤ 커밋 전에 판정 이력 기록 ⑥ 수정과 이력을 같은 커밋으로 push.
안전장치	기준 3종을 바꾸는 지적은 반영하지 않고 사람에게 넘긴다. 이미 기각한 지적은 다시 다루지 않는다. 판정 이력은 덧붙이기만 한다. 지적되지 않은 코드는 건드리지 않는다.
실적	판정 이력 2건 (decisions 파일) — 퇴직금 정산 PR: 4라운드 22건(반영 14 · 기각 7 · 불필요 1). 하네스 정비 PR: 6라운드 35건(반영 33 · 사용자 확정 1 · 기각 1). Codex 리뷰를 거친 PR 140건.

3.4 착수 전 점검 — 컨벤션 탐정

문제	에이전트가 기존 코드 1곳만 보고 "이게 기존 패턴"이라 단정했다. 실제 다수파는 반대 방식으로 10곳 이상 있었고, 다시 작업했다.
동작	신규 API·서비스를 만들기 전 자동 호출. 7개 영역(컨트롤러, 서비스 시그니처, 저장소, 엔티티, DTO, 에러 코드·트랜잭션, 직원 코드 추출)마다 최근 코드 3건 이상을 뽑아 다수파·소수파를 보고한다.
안전장치	표본 3건 미만이면 "판단 불가"로 보고하고 사람이 정한다. 읽기 도구만 가진다.
실적	호출 25회 (작업 기록). 실패 사례 도감에 원인 사건 2건을 기록해 같은 실수를 막는다.

3.5 문서와 코드의 정합 — 문서 동기화 · 정합 점검 · 플로우 시각화

문제	기능명세서가 코드와 어긋나면, 그 문서를 읽고 일하는 에이전트도 틀린다.
동작	문서 원본은 Notion 하나. 로컬은 읽기 전용 캐시이고 수정은 게시 스크립트로만 한다. 정합 점검은 문서 섹션마다 점검 에이전트를 동시에 띄워 "문서에만 있음 / 코드에만 있음 / 서술 불일치"를 파일·라인 근거로 보고한다. 플로우 시각화 에이전트는 기능 하나의 API·계층 흐름·엔티티·권한·에러 코드를 한 장의 문서로 만든다.
안전장치	항상 문서를 코드에 맞춘다. 코드가 의심되면 고치지 않고 보고한다. 애매하면 "확인 필요"로 분류한다.
실적	원천 동기화 문서 262개(명세 55 · 아키텍처 41 · 매뉴얼 49 · 인사이트 40 · 이슈 35 · 테스트 시나리오 14), Notion 페이지 매핑 242개. 플로우 시각화 문서 3건.

3.6 사내 문의 분류 – 문의 트리아지 + 도메인별 읽기 전용 계정

문제	현장에서 올라오는 문의를 개발자가 일일이 열어 보고, 백엔드 문제인지부터 가려야 했다.
동작	미처리 문의 수집 → "백엔드에서 처리 가능한가" 하나의 기준으로 판정(애매하면 사람에게 질문) → 명세·코드 추적과 해당 도메인 계정으로 실데이터 확인 → 처리 상태만 갱신, 분석은 로컬 기록.
안전장치	운영 DB는 도메인별 계정 8종으로 권한 분리 – 잘못 분류하면 DB가 거부하고, 그 거부가 재분류 신호가 된다. 현장 사용자에게 보이는 답변 칸과 "완료" 상태는 에이전트가 쓰지 않는다. 감사 로그는 덧붙이기만 한다.
실적	파일럿 2건 완주(둘 다 백엔드 대상 판정), 감사 로그 6건. 도메인 계정별 허용 테이블: 인사 35 · 공사 41 · 근태 10 · 자재 8 · 원가 8 · 시스템 7 · 급여 4 · 안전 4.

3.7 머지 전 검증 – 기능 검증 · 테스트 코드 규약

문제	컴파일이 통과해도 명세와 다른 API가 머지될 수 있었다.
동작	변경된 API를 추출해 명세 기준으로 정상 · 검증 실패 · 권한 위반 3가지 이상을 실제 호출로 확인하고, 프론트 호출부와 대조해 머지 적합 판정을 낸다. 테스트 코드는 도메인 단위 → 서비스 단위 → 컨트롤러 슬라이스의 계층 규약을 따른다.
안전장치	실패 1건이라도 있으면 부적합. 문서 없이 검증하지 않는다. 쓰기 검증은 로컬의 테스트 현장에서만.
실적	테스트 파일 114개, 테스트 시나리오 문서 14개, 테스트 규약 제정 이후 테스트 커밋 17건.

3.8 세션 하네스 – 훅과 브리핑

훅	실행 시점과 역할
보호 브랜치 차단	명령 실행 전. dev·main 직접 push, main 대상 PR, PR 머지를 차단한다. 우회 옵션도 잡는다.
DB 마이그레이션 번호 충돌	push 전. 다른 브랜치와 번호가 겹치면 차단, 역순이면 경고.
하네스 지도 불일치	push 전. 지도가 실제 자산과 다르면 차단 – 자산을 저장소 밖(S3)으로 옮긴 뒤 유일한 강제 장치.
세션 시작 브리핑	세션 시작·초기화·압축 시. 필독 규칙을 주입하고 규칙 버전이 다르면 경고, Notion 문서를 변경분만 동기화한다.

04 시스템 구성

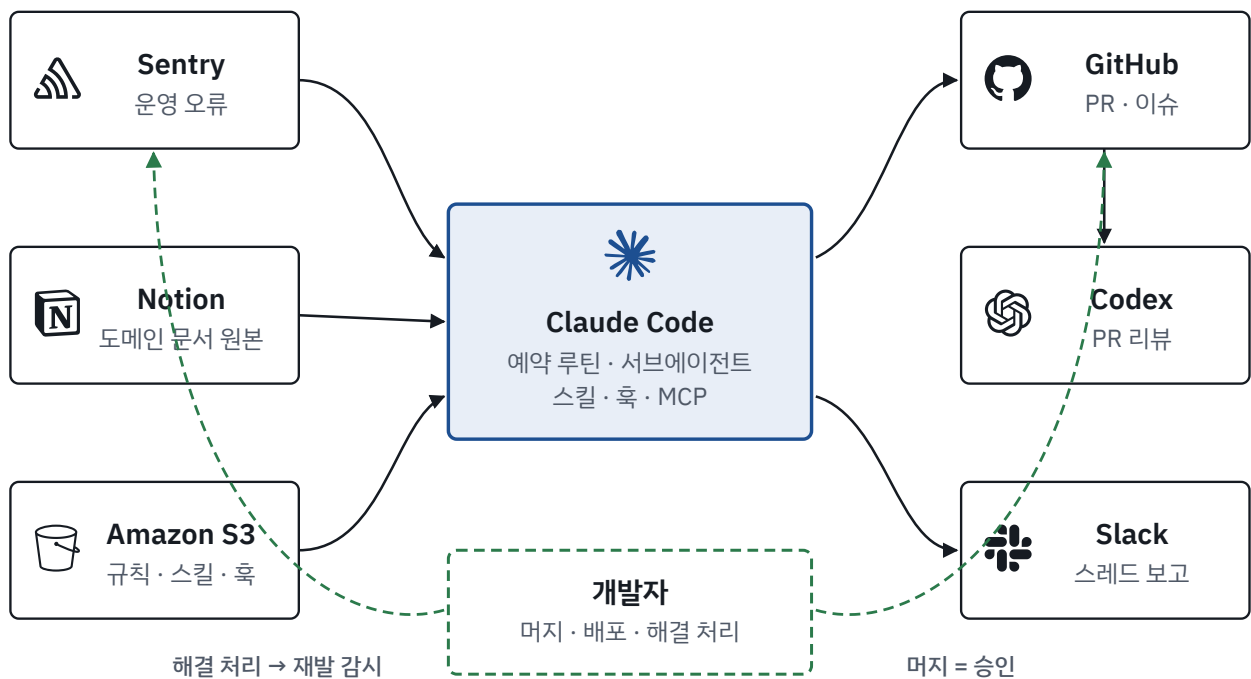


그림 7. 실선은 에이전트, 점선은 사람. 에이전트는 운영 DB·운영 API에 쓰지 않는다.

05 지표

항목	값	세는 기준
하네스 노드 / 연결	47 / 110	하네스 지도 자동 생성 결과
핫픽스 자동 산출 (코드 PR / 근거 이슈)	1 / 11	GitHub 라벨
V1 → V2 API 이관 커밋	13	커밋 제목 분류 (머지 제외)
리뷰 판정 이력 (지적 건수)	57	판정 이력 2개 파일 합계 (22 + 35)
Codex 리뷰를 거친 PR	140	GitHub 검색
컨벤션 탐정 호출	25	작업 기록 (이 PC에 남은 것만)
원천 동기화 문서 / Notion 매핑	262 / 242	파일 수 / 매핑 키
도메인별 읽기 전용 DB 계정	8	권한 스크립트
테스트 파일 / 시나리오 문서	114 / 14	파일 수

2026.10.06 측정. 작업 기록 기반 호출 수는 클라우드 실행과 다른 PC 기록을 포함하지 않는 최소치입니다.