에이전트 플랫폼 운영과 조직 AX 전환

Linux 인프라에서 메모리 · 디스패치 · 개발 루프 · 문서 전파까지

김정한 (JUNGHAN KIM)

역량 및 성과 기술서

한 줄

에이전트 도구를 사용한 사람이 아니라, **에이전트 플랫폼을 운영하고 그 위에 자기 계층을 쌓은 사람입니다.** Linux 인프라 위에서 런타임을 운영하며 메모리, 디스패치, 검토 가능한 개발 루프, 그리고 사람이 이어받을 문서 체계를 만들었습니다.

주요 업무

AX 전환을 위한 시스템 인프라 구축 · 관리와 서비스 설계 · 개발 · 운영. **운영이 먼저 있었고, 계층은 그 다음에 나왔습니다.**

- 에이전트 런타임(OpenClaw)을 2026년 2월부터 6월까지 **약 5개월, 20회 이상의 버전 사이클로 운영했습니다.** Oracle Cloud ARM(aarch64) · Docker · 메신저 채널 다중 연동.
- 업스트림은 사실상 **1인이 유지하는 프로젝트입니다.** 우리 조직에 업스트림 담당자가 없다는 뜻이고, 그래서 릴리즈의 의미를 내 저장소 관점으로 번역하는 일 자체가 운영이 됩니다.
- 회귀가 나면 최신판을 고집하지 않습니다. 응답 지연이 터졌을 때 게이트웨이가 단일 스레드에서 **CPU 102%로 회전하고,** 부팅이 정상 11초에서 **88초까지 늘어난 것을 계측했습니다.** 클린 부팅에서도 재현시켜 설정 오류 가설을 배제한 뒤 마지막 정상 판으로 롤백했습니다.
- 다음 인시던트는 업스트림 릴리즈 노트에서 **내 증상의 정공법 수정을 식별해 해결했습니다.** 부팅 45.4초에서 5.8초로, 메모리 816MiB에서 246MiB로 내려갔습니다.
- 재발을 막은 것은 패치가 아니라 **규칙입니다.** 두 버전 건너뛰기 금지, 비프로덕션 에이전트에 24시간 이상 스테이지한 뒤 승격, `stuck session` 한 줄도 배포 중단 사유, 그리고 응답성이 SLO.
- 이 운영 경험은 세 개의 독립 계층으로 분리되었습니다 — andenken(메모리), entwurf(디스패치), forge-config(검토 가능한 개발 루프).
- 회사에서는 GPU · 컨테이너 · 문서 임베딩 · 데이터 파이프라인을 포함한 AI 인프라를 구축하고, 이미 운영 중인 업무 시스템에 **읽기 전용 에이전트 접점을 붙였습니다.**

판단의 규율 — 도구가 시키는 대로 하지 않습니다. 진단 도구의 보안 권고를 우리 배포 맥락에서 검토해 거짓 양성으로 판정하고 적용하지 않았습니다. 설정을 재작성하는 자동 수정 옵션도 쓰지 않습니다. AX 도입이 실패하는 가장 흔한 방식이/도구의 권고를 이유도 모른 채 따르는 것/이고, 그에 대한 반례를 운영 기록으로 갖고 있습니다.

AI 관련 가이드 · 위키 작성 · 배포와 교육.

- **업스트림 변경을 내 저장소별 영향으로 번역하는 문서를 유지합니다.** 릴리즈 노트를 그대로 옮기지 않고 "어느 리포의 무엇을 봐야 하는가"라는 신호로 바꿉니다.
- 문서를 세 층으로 나눕니다. `AGENTS.md` 는 영속 기준선, `NEXT.md` 는 세션 간 인계, 스킬 문서는 실행 계약입니다. **사람이든 에이전트든 같은 규칙을 읽습니다.**
- **40개 이상의 에이전트 스킬이 그 계약을 소비합니다.** 문서가 장식이 아니라 실행되는 인터페이스라는 뜻입니다.
- 공개 디지털 가든과 라이브 agenda로 배포까지 갑니다. 작성에서 끝내지 않고, 다시 읽히는 표면으로 만듭니다.

Author's Contact Information: 김정한 (Junghan Kim).

- 독자에 따라 표면을 나눕니다. 개발자에게는 재현 명령과 불변식을, 비개발자에게는 일일 판독과 근거 링크를 줍니다.

자격 요건

Linux 환경의 개발과 운영. NixOS로 노트북·서버·ARM 장치를 선언적으로 관리합니다. 컨테이너 서비스, GPU 워크로드, Yocto 임베디드 Linux, glibc와 musl, ARM과 RISC-V를 같은 재현성 원칙으로 다룹니다. **환경이 재현되지 않으면 운영 판단도 재현되지 않습니다.**

Claude Code·*Codex* 등 *Agentic AI*를 활용한 개발. Claude Code와 Codex를 매일 씁니다. 그러나 이 항목의 답은 사용량이 아닙니다. 하네스가 세션을 넘어 위임·연속성·공유 도구를 보존하도록 entwurf를 만들었고, 40개 이상의 스킬을 설계·운영했습니다. **도구 사용자가 아니라 하네스 운영자입니다.**

백엔드 시스템 개발과 운영. Go와 Clojure로 API, 단일 바이너리 서비스, IoT 백엔드, 데이터·운영 CLI를 만들었습니다.

경계: Java/Spring 직접 경험을 Go/Clojure 경험으로 바꾸어 쓰지 않습니다. JVM과 GraalVM은 Clojure 서비스의 빌드와 운영에서 다룹니다.

개발·비개발 영역의 *AX* 전환. 대시보드를 하나 더 만들지 않습니다. **도메인의 주인에게 에이전트를 붙입니다.** 운영팀이 실제 상담 데이터를 자연어로 묻고 근거를 개별 대화까지 역추적하는 워크벤치, 장애 증거를 살아 있는 원본에서 회수하는 읽기 전용 워크벤치가 그 사례입니다. 모든 시각은 하나의 시간축으로 정규화하고, 수치는 원 이벤트까지 되짚을 수 있게 둡니다.

우대 사항

*Web Framework*와 서비스 표면. 문제에 따라 Go HTTP 서비스, Clojure 서버 렌더링, 정적 웹, React/Preact/Lit 컴포넌트를 선택했습니다. **프레임워크 이름보다 배포 단위와 운영 경계를 먼저 정합니다.**

DB·*MQ*·*VM*·컨테이너. PostgreSQL JSONB, pgvector, SQLite, LanceDB, MQTT, AWS IoT, Oracle ARM VM, Docker와 GPU 컨테이너를 실제 시스템에서 다뤘습니다.

LLM·*RAG*. andenken은 메모리를 **세 축으로 가릅니다** — 응답 전 차단형 리콜, 임베딩 검색, 야간 통합. 일반적인 RAG 스택과 다른 지점이 여기입니다. 현재 구현된 축은 임베딩입니다: LanceDB, Qwen3-Embedding-8B(4096차원), **벡터와 BM25 하이브리드에 점수 정규화,** 한↔영 교차 질의 확장, 그리고 회수(recall) 추적.

회사에서는 임베딩·리랭킹 서빙과 업무 문서 파이프라인을 운영했습니다.

경계: 파인튜닝 중심이 아닙니다. 정직하게 밝힙니다. 강점축은 **임베딩 서빙, 리랭킹, 검색 품질 최적화, 프롬프트·체인 설계**라는 실서비스 최적화 축입니다.

*BMAD*와 구조화된 *AI Workflow*. **BMAD 직접 사용 경험은 없습니다.** 대신 그 방법론이 다루는 문제층 — 역할 분리, 구조화된 계획, 컨텍스트를 실은 인계, 규모별 깊이 조절 — 을 `AGENTS/NEXT` 문서 계약, entwurf의 위임, forge-config의 리뷰 게이트로 독립적으로 구현했습니다. 동일하다고 주장하지 않고, **차이를 설명할 수 있는 구현을 제시합니다.**

클라우드. AWS IoT, Oracle Cloud ARM, Cloudflare Zero Trust, Netlify에서 서비스를 개발·운영했습니다.

Frontend. React 19와 `@a2ui/react` 기반 A2UI 뷰어, Preact/TSX 가든 컴포넌트, Lit 웹컴포넌트를 구현했습니다. **같은 A2UI 개념을 서로 다른 렌더러 위에 두 번 구현했다는 것이 실제 증거입니다** (Lit / React).

경계: Tailwind 직접 사용 경험은 없습니다. SCSS 변수와 디자인 토큰을 사용했습니다.

협업과 커뮤니케이션. 역할과 파일 경계를 먼저 나눠 병렬 에이전트 작업에서도 충돌을 막고, 대화를 durable issue와 리뷰 가능한 기록으로 굳힙니다. 회사에서는 운영·제품·개발 조직이 **이어받을 수 있는 형태로 도구를 넘겼습니다.**

개발자 · 비개발자 대상 문서. **같은 데이터라도 독자의 행위가 달라지면 문서 표면도 달라져야 합니다.** 개발자에게는 재현 명령과 불변식을, 비개발자에게는 일일 판독과 근거 링크를 제공합니다.

검증 가능한 기록

공개 표면

주장	확인할 곳
에이전트 디스패치와 연속성	entwurf
세션 · 지식 메모리	andenken
검토 가능한 개발 루프	forge-config
임베디드와 엣지 AI	homeagent-config
재현 가능한 Linux 기반	nixos-config
공개 지식 · 문서 체계	Digital Garden
라이브 작업 · 통계 표면	Agenda · /api/stats
제3자의 확장	entwurf #40
제3자가 받아들인 코드	ghostel #343 · #510

마지막 두 줄이 **수용(reception)의 증거입니다.** 자기 저장소는 능력을 보여주지만, 저자가 유일한 심판인 닫힌 루프를 의심하는 독자에게는 답이 되지 않습니다.

시간축

사람이 손으로 기록한 생활과 저널, 에이전트가 남긴 stamp, 실제 커밋과 노트를 하나의 KST 축에서 읽습니다. 원본(FULL)은 제목과 참조를 품으므로 **공개하지 않습니다.** 아래는 allowlist로 집계한 공개 판독이고, `timeline/project.py` 가 FULL과 스냅샷의 짝(바이트 지문)을 검증한 뒤에만 생성합니다. 같은 입력이면 같은 바이트가 나오고, 날짜는 하나도 손으로 적히지 않았습니다.

에이전트를 수십 개 붙여도 지나간 시간은 만들어낼 수 없습니다. 이 축은 산출물이 아니라 하루를 셉니다. 같은 기간을 어느 깊이까지 읽느냐에 따라, 살았지만 아무 산출물도 남기지 않은 날이 보이거나 사라집니다.

깊이	무엇이 보이나	기록의 성격	커버리지
0	산 대로의 하루 — 수면 · 가족 · 독서 · 본짓	의도	194일 / 194일
1	그날 손으로 적은 것	의도	133일 / 194일
2	에이전트가 남긴 스탬프	잔여물	136일 / 194일
3	커밋과 노트	잔여물	187일 / 194일

렌즈를 바꾸면 며칠이 사라지는가. 창은 2026-01-01 이상 2026-07-14 미만(KST)이고, 그 사이의 모든 날 194일이 분모입니다.

읽는 렌즈	보이는 날	사라지는 날
깊이 2 · 3 — 잔여물만 (에이전트 스탬프 · 커밋 · 노트)	192일	2일
깊이 1 · 2 · 3 — 잔여물 + 손으로 적은 저널	193일	1일
깊이 0 포함 — 산 대로의 하루	194일	0일

잔여물이 통째로 침묵한 날. 커밋도 노트도 에이전트 스탬프도 없는데, 산 시간은 있습니다. 산출물만 읽는 판독기에게 이 날들은 공백입니다.

2026-02-07 가족 611.6분 · 수면 484.6분 · 독서 358.6분 (합 1,455분) — 저널 0건 · 스탬프 0건 · 커밋 · 노트 0건
2026-07-11 수면 504.0분 · 독서 477.2분 · 가족 408.3분 (합 1,390분) — 저널 2건 · 스탬프 0건 · 커밋 · 노트 0건

이 판독의 출처. 인용은 내용 지문 하나로 하지 않습니다 — 같은 내용이라도 창이 다르면 다른 판독이고, 소스가 부분적이면 다른 품질이며, 깊이 0을 만든 도구가 다르면 다른 손이 적은 것입니다.

창 (`as_of`) 2026-07-14T00:00:00+09:00 — 배타적 상한

소스 상태 git ok(9,776) · note partial(5,511) · agenda ok(3,682) · journal ok(1,292) · timelog ok(8,400)

깊이 0 도구 (`src_tree`) lifetract `bbcc7d483f4537b9 5d130e1c356c57d1 31353a71`

이벤트 28,661건 / 엔티티 23,865건

관측 지문 `content_sha256`

`11be45396ace7834 37e1f393d7dc36cd f455332977117322 e3d39da6d8237f7b`

수집기 `code_sha256` — collector 0.7.0

`a847831b2edd8294 6279753e1ec5b3a0 77a50d83a01e2dd5 0308a99ffb6f3ca4`

이 지문이 증명하지 않는 것. 해시는 바이트와 내용이 같은지, 바꿔치기가 없었는지를 확인할 뿐입니다. 이 코드가 실제로 돌았음도, 소스가 진실하고 완전함도 증명하지 않습니다. 깊이 0은 폰에서 손으로 export한 1인칭 기록이라 외부에서 원본을 재구성할 길이 없고, 확인할 수 있는 것은 파생의 일관성입니다. 원본(FULL)은 제목과 참조를 품으므로 공개하지 않습니다 — 이 문서에는 집계만 나옵니다. 시간 블록은 시작 날짜에 귀속되며, 하루를 24시간 파티션으로 나눈 것이 아닙니다.

수치의 원칙

큰 숫자는 장식이 아니라 더 깊은 기록으로 들어가는 입구입니다. 공개 통계는 라이브 endpoint와 함께 제시하고, 프로젝트 수치는 기간 · 포함정책 · 검증 링크를 붙입니다.

경계

주장을 키우는 것보다 **어디까지가 내 것이 아닌지 먼저 말하는 것이 신뢰를 만든다고 봅니다.**

- Java/Spring 직접 경험을 Go/Clojure 경험으로 바꾸어 쓰지 않습니다.
- BMAD를 사용했다고 쓰지 않습니다. 같은 문제층의 독립 구현과 그 차이를 보여줍니다.
- Tailwind 직접 사용 경험은 없습니다. React/Preact/Lit과 디자인 토큰 경험을 제시합니다.
- 파인튜닝 중심이 아닙니다. 임베딩 서빙, 리랭킹, 검색 품질, 프롬프트 · 체인이 강점축입니다.
- OpenClaw는 상류 소프트웨어입니다. 제 것은 그 위의 운영층 · 메모리 계층 · 개발 루프이고, 이 구분을 흐리지 않습니다.
- 회사 코드와 식별 가능한 운영 데이터는 공개하지 않습니다. 공개 저장소 옆에서 구조와 판단만 설명합니다.
- 해시는 입력과 산출물의 동일성을 확인할 뿐, 원천의 진실성이나 완전성을 대신 증명하지 않습니다.