

MSAP.ai

msap.ai

Hermes Agent 소개

오픈소스 AI Agent 플랫폼

자기 개선형 오픈소스 에이전트의 구조 · 기능 · 도입 방법

PROJECT CODE

HERMES-AGENT-V2

DATE

2026.10.31

8개 파트 · 22편 구성

PART 1 배경

PART 2 개요

PART 3 아키텍처

PART 4 핵심 기능

PART 5 On-Premise

PART 6 비교

PART 7 사례

PART 8 보안 · 도입

CONFIDENTIAL

Hermes Agent 학습 순서

8개 파트 · 22편 · 중분류 하나가 영상 한 편

PART 1 AI Agent 와 등장 배경

- 1-1 챗봇과 에이전트의 실행 방식 차이
- 1-2 에이전트 도입 배경과 업무 과제

PART 2 Hermes Agent 개요

- 2-1 Hermes Agent 정의와 프로젝트 현황
- 2-2 Hermes Agent 설계의 네 축

PART 3 Hermes Agent 아키텍처

- 3-1 판단과 실행을 나눈 Hermes 구조
- 3-2 Gateway 와 메시징 채널 구성
- 3-3 Profile 과 Kanban 의 역할 분담

PART 4 Hermes Agent 핵심 기능

- 4-1 Skill 시스템의 구조와 작성 방식
- 4-2 Memory 와 학습 루프
- 4-3 Cron 과 무인 실행
- 4-4 MCP 와 LiteLLM 기반 외부 연동

PART 5 On-Premise AI

- 5-1 로컬 LLM 이 필요한 업무와 이유
- 5-2 로컬 모델과 하드웨어 선택

PART 6 AI Agent 비교

- 6-1 오픈소스 에이전트와 Hermes 비교
- 6-2 에이전트 도입 형태 선택 기준

PART 7 적용 사례

- 7-1 업종별 적용과 에이전트 활용 유형
- 7-2 사례: 로컬 LLM 기온 감시 에이전트
- 7-3 MSAP.ai 사내 Hermes 운영 사례

PART 8 보안과 도입

- 8-1 에이전트 위험과 기능 요구
- 8-2 Hermes Agent 보안 기능
- 8-3 에이전트 가드레일과 운영 원칙
- 8-4 Hermes Agent 도입 로드맵

PART 1

AI Agent 와 등장 배경

EPISODE 1-1

챗봇과 에이전트의 실행 방식 차이

AI Agent 정의와 LLM · RAG · RPA 와의 구분

이 편에서 다루는 내용

- 1 AI Agent 의 정의와 동작 순환
- 2 챗봇과 에이전트의 실행 방식 비교
- 3 기존 AI 기술과 Agent 의 차이
- 4 에이전트는 권한을 가진 자동화



AI Agent 의 정의와 동작 순환

1-1 · 챗봇과 에이전트의 실행 방식 차이

KEY INSIGHT

AI Agent 는 목표를 받아 관측 · 분석 · 계획 · 실행을 반복함 —
사람이 단계마다 지시하지 않아도 목표에 닿을 때까지 일이 이어짐

정의 LLM 을 판단 엔진으로 삼아 환경을 인식하고, 외부 도구를 써서 목표를 이룰 때까지 스스로 행동하는 자율 시스템



AI Agent 의 핵심 특성 4가지



자율성

Autonomy

단계마다 지시를 기다리지 않고 다음 행동을 스스로 정함



도구 사용

Tool Use

검색 · 파일 · API 같은 외부 도구로 실제 작업 수행



계획과 추론

Planning & Reasoning

목표를 하위 작업으로 나누고 순서를 세움



목표 지향

Goal-oriented

요청 하나가 아니라 목표 달성 여부로 끝을 판단



◆ KEY INSIGHT

에이전트는 파일 변경 · 외부 호출 · 메시지 발송처럼 시스템에 남는 실행을 함 —
위임한 동작 목록의 폭이 곧 걸어야 할 통제의 범위를 정함



챗봇

응답형 도구 · 사람이 매 단계 지시



에이전트

실행형 도구 · 목표만 받고 실행을 이어 감



실행 단위

요청 하나에 응답 하나로 종료

목표 하나로 여러 단계를 연속 수행



다음 단계 결정

언제나 사람이 판단하고 지시

도구 선택과 중단 시점까지 스스로 판단



실행이 남기는 것

대화창에 표시된 텍스트

파일 변경 · 외부 호출 · 메시지 발송



세션 간 기억

매 세션 처음부터 다시 설명

메모리와 스킬로 다음 세션에 이어짐



되돌리는 비용

낮음 — 출력을 버리면 끝

높음 — 시스템에 상태가 이미 남음



◆ KEY INSIGHT

Agent 는 실행 결과를 보고 다음 행동을 스스로 정함 —
정해진 흐름을 벗어난 업무까지 맡길 수 있다는 점이 기존 기술과 다름

기술	하는 일	하지 못하는 일 · 한계
LLM	프롬프트에 따라 텍스트 생성	행동 실행 · 환경 정보 취득
RAG	검색한 문서를 넣어 답변 생성	결과에 따라 검색을 이어 가기 · 검색 방향 조정
RPA Workflow	정해진 업무 절차를 자동 실행	환경 변화 적응 · 업무마다 흐름을 새로 만들어야 함
AI Agent	의도 이해 · 계획 · 행동 · 결과에 따른 적응	권한 · 검증 설계가 없으면 실행 사고로 이어짐

용어 구분



AI Agent

하나의 과업을 수행하는 개별 실행 단위 — LLM + 도구 + 계획



Agentic AI

여러 Agent · 도구 · 데이터를 조율하는 시스템 구조 — 오케스트레이션 계층



◆ KEY INSIGHT

에이전트의 잘못된 판단은 실행된 뒤에 발견됨 —
편의 기능과 같은 무게로 가드레일을 설계할 대상이 됨

챗봇 — 텍스트 생성

- **출력물** 문장 · 코드 조각 · 요약
- **영향 범위** 화면 안에서 끝남
- **잘못된 경우** 사용자가 버리면 종료
- **되돌리는 비용** 사실상 없음



에이전트 — 시스템 조작

- **출력물** 파일 변경 · API 호출 · 배포 · 발신
- **영향 범위** 실제 시스템과 외부 수신자
- **잘못된 경우** 이미 실행된 뒤에 발견
- **되돌리는 비용** 크고 때로는 불가

그래서 필요한 것 — 실행 권한을 감싸는 네 겹

권한 최소화

필요한 도구 · 채널 · 저장소만 허용
목록으로 허가

위험 조작 승인

삭제 · 강제 반영 · 외부 발신 · 과금은 사람
승인 후 실행

경계 분리

채널 · 역할별로 기억과 권한을 나눠 오염
확산 차단

실행 추적

누가 무엇을 읽고 실행하고 보냈는지
로그로 남김

PART 1

AI Agent 와 등장 배경

EPISODE 1-2

에이전트 도입 배경과 업무 과제

운영 영역 확산 · 반복 업무 · 유지보수 부담 관점

이 편에서 다루는 내용

- 1 에이전트 적용 범위의 확산
- 2 소수 개발 조직의 세 가지 병목
- 3 정적 에이전트의 유지보수 부담
- 4 반복 업무 위임의 네 가지 구성 요소



◆ KEY INSIGHT

2026년 에이전트는 운영 시스템을 직접 조작하기 시작함 —
어떤 제품을 고르느냐보다 권한 · 승인 · 감사 체제가 도입 성패를 가름

2025

AI 에이전트 원년



- 개발 보조 중심의 실험
- 코드 작성 · 리뷰 지원
- 개인 단위 도입이 대부분



2026

운영 영역으로 확산



- 시스템 운영에도 에이전트 보급
- 인시던트 대응 · SRE 자동화
- 조직 단위 도입 논의 시작



다음 과제

체제 설계



- 권한 · 승인 · 감사 정책
- 사람이 개입할 지점 정의
- 실패 시 되돌리는 절차

시사점 — 도구 선택보다 체제 설계가 먼저

개발 보조 단계에서는 잘못된 출력을 사람이 버리면 끝났지만, 운영 단계의 에이전트는 실제 시스템을 조작함
따라서 어떤 제품을 고르느냐보다 어떤 권한과 승인 구조로 감싸느냐가 도입 성패를 결정함



◆ KEY INSIGHT

개발 시간 부족의 원인은 개발 밖의 일에 있음 —
채용은 즉시 되지 않아 일을 맡길 대상을 새로 정하는 편이 더 빠름



CHORES

잡무로 녹는 시간

문의 대응 · 회의 · 최신 정보 캐치업 · 근태 입력 ·
자료 작성이 쌓일수록 개발 시간이 깎임

01



CONTEXT

사양 확인의 누적

기능 개발 전 조사와 "이 사양 어떻더라?" 같은 확인
의뢰마다 코드를 다시 추적함

02



HEADCOUNT

한 명만 더 있으면

시간도 예산도 없고 채용은 즉시 되지 않음 — 인력
부족이 그대로 남음

03

그래서 내린 결론

도구 도입이 아니라 인력 배치의 문제로 재정의

사람을 늘리는 대신 개발이 아닌 일을 맡을 대상을 새로 정함



AI 를 신입사원처럼 배치



◆ KEY INSIGHT

기억 없는 에이전트는 세션마다 같은 지시를 다시 받음 —
운영 대수가 늘수록 프롬프트 수정 · 재배포 비용이 자동화 이득을 깎아 먹음

정적 에이전트의 약점

맥락 상실

세션이 끝나면 환경과 절차를 모두 잊음 — 매번 다시 지시해야 함

조용한 실패

대상 화면 구조가 바뀌면 알림 없이 멈춤 — 사람이 늦게 알아챌

끝나지 않는 유지보수



프롬프트 미세 조정에 쫓겨 자동화 이득이 상쇄됨

필요한 것

스스로 배우는 에이전트

- 실패에서 배워 절차를 고침
- 세션이 끝나도 기억을 유지
- 바뀐 환경을 다음 실행에 반영
- 사람은 결과만 검토

→ Hermes Agent 가 겨냥하는 문제



반복 업무 위임의 네 가지 구성 요소

1-2 · 에이전트 도입 배경과 업무 과제

◆ KEY INSIGHT

절차 기억 · 채널 · 무인 실행 · 검증 중 하나만 빠져도 사람 손으로 돌아감 —
다만 업무 성격에 따라 먼저 갖출 요소가 달라 네 가지를 같은 수준으로 맞출 필요는 없음

📄 절차 기억

같은 일을 다시 설명하지 않도록 수행
절차를 저장

빠지면

요청마다 맥락을 다시 설명 — 위임 효과
소멸

💬 채널

담당자가 자리에 없어도 요청이
들어오는 입구

빠지면

근무 시간에만 요청 수신 — 상시 처리
불가

⚡ 무인 실행

단계마다 개입 없이 목표까지 실행을
이어 감

빠지면

사람이 단계마다 확인 — 자동화 효과
없음

✅ 검증

실행이 끝난 뒤 결과가 맞는지 확인하는
절차

빠지면

확인 없는 결과가 다음 업무의 입력으로
넘어감

업무마다 달라지는 비중

하루 한 번 결과만 보는 정산 업무

검증

검증 비중이 크고 무인 실행 비중은 작음

야간 요청을 아침까지 처리하는 운영 업무

채널 · 무인 실행

채널과 무인 실행이 핵심 요건

PART 2

Hermes Agent 개요

EPISODE 2-1

Hermes Agent 정의와 프로젝트 현황

개발 주체 · 성장 지표 · 라이선스 · 설계 위치

이 편에서 다루는 내용

- 1 Hermes Agent 한 문장 정의
- 2 숫자로 보는 Hermes Agent 현황
- 3 MIT 라이선스의 사내 도입 의미
- 4 Harness 구성 요소와 Hermes 기능 대응



◆ KEY INSIGHT

Hermes Agent의 가치는 운영 기간에 비례해 커짐 —
첫날 성능보다 3개월 뒤 얼마나 정교해졌는지로 평가할 대상임

DEFINITION

오픈소스 자율 AI 에이전트 — 쓸수록 자기 업무에 맞게 정교해짐

자기 개선 · 메모리 관리 · 저비용 · 안전한 실행을 하나의 설계에 통합



개발 주체

Nous Research

오픈소스 공개 — 자체 호스팅과 내부 개조 가능



성격

자기 개선형 실행 기반

경험에서 스킬과 기억을 키우는 에이전트 런타임



운용 형태

24시간 상주

서버 · 클라우드 · 사내 장비에서 자기 개선 루프를 지속



위치

특정 업무를 키우는 기반

좁은 영역에서 오래 돌며 판단을 축적하는 용도

겨냥하지 않는 용도

❌ 범용 상담 어시스턴트

아무 질문에나 답하는 만능 상담역은 목표가 아님

❌ 노코드 워크플로 도구

화면에서 블록을 잇는 손쉬운 구축 방식은 제공하지 않음

✅ 평가 기준

첫날 성능이 아니라 3개월 뒤 정교함



숫자로 보는 Hermes Agent 현황

2-1 · Hermes Agent 정의와 프로젝트 현황

◆ KEY INSIGHT

2026년 3월 첫 릴리스 뒤 44회 릴리스와 약 26만 스타를 기록함 —
변화가 빠른 만큼 사내 도입 때는 업그레이드 전 검증 부담도 함께 커짐

최신 버전



v0.22.5

2026-10-23 게시 · 약 81개 PR 묶음

GitHub 스타



258,989

포크 56,934 · 2026-10-31 조회

GitHub 릴리스



44회

첫 릴리스 v0.2.0 (2026-03-12)

라이선스



MIT

상업 이용 · 수정 · 비공개 재사용 허용

추론 제공자



43개

공식 표 기준 · 로컬 Ollama · vLLM · SGLang · llama.cpp · LM Studio

메시징 플랫폼



20개 이상

공식 비교표 28행 · Telegram · Slack · Teams 등

수치 출처가 문서마다 다름 — 도구 수는 README 40+ · 문서 홈 60+ · 레퍼런스 표 71개로 표기되므로 인용 시 기준을 함께 적음



◆ KEY INSIGHT

MIT 는 수정본 비공개 · 상업 이용을 모두 허용함 —
법무 검토 시간은 짧아지지만 의존 패키지 라이선스는 따로 점검해야 할 대상임

라이선스	상업적 이용	소스 수정	수정본 비공개	특허	주요 의무
MIT (Hermes Agent)	허용	허용	허용	명시 없음	저작권 고지 보존
Apache 2.0	허용	허용	허용	특허 허여 명시	고지 + 변경 사항 명시
GPL v3	허용	허용	불허	특허 허여 명시	수정 소스 공개 의무
AGPL v3	허용	허용	불허	특허 허여 명시	네트워크 서비스도 소스 공개

사내 도입 때 확인할 것

🔒 수정본 비공개

사내에서 고친 코드를 외부에 공개하지 않아도 됨 — GPL · AGPL 은 지식재산 보호와 충돌 가능

📄 고지 의무

외부 배포가 아닌 사내 사용이면 저작권 고지 의무의 실질 영향이 거의 없음

⚠️ 의존 패키지

Synopsys 2025: 코드베이스 60% 에서 라이선스 충돌 — SBOM 기반 점검 권장



◆ KEY INSIGHT

Hermes 는 사전 제어 · 사후 점검 · 맥락 공급을 자체 기능으로 갖추 —
모델이 상품화돼도 모델을 감싸는 구조는 그대로 남아 경쟁력의 근거가 됨

Agent 구성 Model + Harness

모델은 상품화되고, 차이는 모델을 감싸는 실행 구조에서 남음

Guides 사전 제어

Hermes 구현



Skill 과 Profile

- Profile 마다 허용 도구 · 스킬 · 메모리 범위를 미리 정함
- SOUL.md 로 말투와 금지 사항을 고정

Sensors 사후 점검

Hermes 구현



백그라운드 리뷰와 Curator

- 실행이 끝난 뒤 회고해 절차를 스킬로 남김
- Curator 가 쓰지 않는 스킬을 주기적으로 정리

Context Pipelines 맥락 공급

Hermes 구현



세션 검색과 Kanban

- SQLite 에 쌓인 세션 기록을 필요할 때 검색
- Kanban 이 현재 작업 상태와 선행 작업을 공급

PART 2

Hermes Agent 개요

EPISODE 2-2

Hermes Agent 설계의 네 축

자기 진화 · 메모리 · 비용 · 실행 환경 관점

이 편에서 다루는 내용

- 1 축 1. 성공한 절차의 스킬 저장
- 2 축 2. 전량 보관과 선별 주입 메모리
- 3 축 3. 캐시를 살리는 프롬프트 배치
- 4 축 4. 실행 환경 선택의 자유
- 5 일반 챗봇과 Hermes 의 운영 기간별 차이



축 1. 성공한 절차의 스킬 저장

2-2 · Hermes Agent 설계의 네 축

◆ KEY INSIGHT

한 번 성공한 절차가 SKILL.md 로 남아 다음 실행부터 탐색 없이 재사용됨 —
학습 비용은 첫 실행 한 번이고 효과는 운영 기간 내내 쌓임

AXIS 01

자기 진화

성공한 수순을 스킬로 저장하고
다음부터 재사용



쓸수록 정교해짐



1 첫 실행

사람이 지시하고 에이전트가 수행

시행착오가 생기고 토큰과 시간이 많이 듦



2 스킬 저장

무엇을 · 어떤 순서로 · 어떤 조건에서 했는지 기록

재현 가능한 Markdown 파일 (SKILL.md)



3 재사용

같은 상황에서 탐색 없이 바로 실행

같은 지시를 반복할 필요 없음 · 시간과 비용 감소



축 2. 전량 보관과 선별 주입 메모리

2-2 · Hermes Agent 설계의 네 축

◆ KEY INSIGHT

대화는 모두 SQLite 에 남고 프롬프트에는 작은 메모리만 실림 —
24시간 상주해도 매 회차 문맥 비용이 늘지 않아 장기 운영이 가능해짐

메모리를 세 층으로 다룸



보관

모든 세션을 SQLite state.db 에 저장

FTS5 전문 검색 · 질의 약 20ms · 보관 상한 없음



압축

길어진 대화를 요약으로 대체

의미는 남기고 길이는 줄임 — 원문을 그대로 싣지 않음



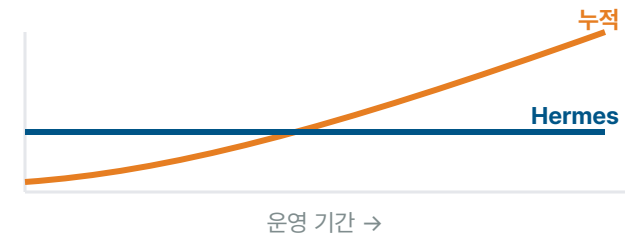
주입

세션 시작 때 작은 메모리만 프롬프트에 실음

MEMORY.md 2,200자 · USER.md 1,375자

단순히 이력을 쌓는 방식

시간이 갈수록 문맥 길이와 비용, 지연이 함께 늘어남



효과

- 기록은 전부 남아 감사 · 추적에 사용
- 매 회차 토큰 사용량이 일정
- 필요한 과거는 검색으로 찾아 씀



축 3. 캐시를 살리는 프롬프트 배치

2-2 · Hermes Agent 설계의 네 축

◆ KEY INSIGHT

규칙 · 인격 · 스킬을 앞에 고정하고 관측값만 뒤에 붙임 —
상주 에이전트일수록 반복 계산이 줄어 비용 절감 폭이 커짐

프롬프트를 쌓는 순서 — 바뀌지 않는 것에서 바뀌는 것으로

시스템 규칙

무엇을 해도 되고 무엇을 하면 안 되는가

인격 파일 (SOUL.md)

어떤 말투와 태도로 말하는가

축적된 스킬

지금까지 학습한 판단 노하우

이번 회차 관측값 · 최근 대화

새로 들어온 데이터와 사람의 최근 지시

앞 블록 · 매 회차 동일

캐시 적중 → 재계산 없음

뒤 블록 · 매 회차 변동

이 부분만 새로 계산



누적 효과

상주 에이전트일수록 절감 폭이 커짐 — 로컬 LLM 에서도 같은 이득



순서를 뒤집으면

가변 정보를 앞에 두면 뒤의 불변 블록까지 매번 다시 계산



축 4. 실행 환경 선택의 자유

2-2 · Hermes Agent 설계의 네 축

◆ KEY INSIGHT

모델 · 채널 · 실행 위치를 조직 제약에 맞춰 고를 수 있음 —
폐쇄망과 기밀 데이터 업무까지 에이전트를 적용할 수 있는 범위가 넓어짐

☁ 모델

어디서 판단할 것인가

로컬 LLM

데이터가 밖으로 나가지 않음 · 과금 없음

클라우드 API

공식 표 43개 제공자 · 최고 성능 모델

💬 채널

어디에 상주할 것인가

사내 메신저

팀 전체가 함께 사용

대시보드 · 외부 알림

20개 이상 메시징 플랫폼 중 선택

🏠 실행 환경

어디서 명령을 돌릴 것인가

터미널 백엔드 6종

local · docker · ssh · modal · daytona · vercel_sandbox

격리 실행

에이전트가 닿는 범위를 물리적으로 제한

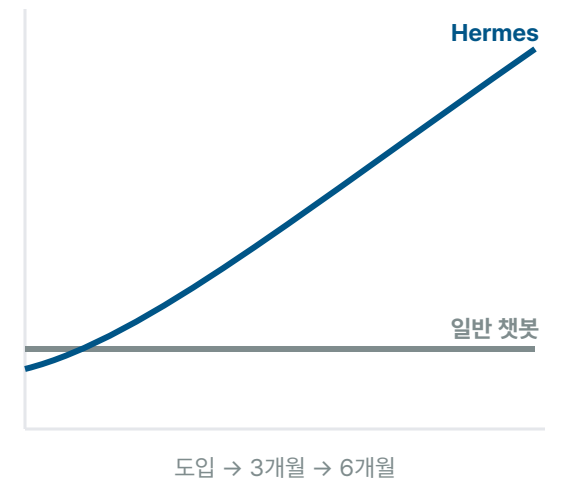


◆ KEY INSIGHT

운영 기간이 길수록 재설명 부담이 줄고 판단 근거가 쌓임 —
짧은 PoC 만으로는 효과를 판단하기 어려워 평가 기간을 길게 잡아야 함

비교 항목	일반 LLM 챗봇	Hermes Agent
세션 간 맥락	매 세션 초기화	메모리로 유지
반복 작업 학습	없음	성공한 절차를 스킬로 자동 저장
운영 기간 효과	첫날과 같음	사용할수록 업무에 맞게 개인화
재설명 부담	매번 발생	점차 감소

운영 기간별 유용성



네 축이 함께 만드는 차이

1 자기 진화

2 메모리 관리

3 캐시 비용 설계

4 선택의 자유

PART 3

Hermes Agent 아키텍처

EPISODE 3-1

판단과 실행을 나눈 Hermes 구조

LLM · 런타임 · 인격 · 지식의 역할 분리

이 편에서 다루는 내용

- 1 LLM 과 Agent 의 역할 분리
- 2 교체 가능한 축과 고정된 코어
- 3 SOUL.md 와 skills 의 분리
- 4 skills 파일의 성장 단계



LLM 과 Agent 의 역할 분리

3-1 · 판단과 실행을 나눈 Hermes 구조

KEY INSIGHT

LLM 은 판단만, Agent 는 실행만 맡음 —

경계가 분명해 모델을 바꿔도 루프 · 도구 · 이력 코드는 그대로 유지됨



판단 레인 — LLM

결정만 하고 아무것도 실행하지 않음

도구 호출 결정

언제 · 어떤 도구를 · 어떤 인자로

가치와 긴급도 판정

알릴 만한 변화인지, 긴급인지

최종 문면 작성

인격과 스킬 파일에 따라



결정만 전달



실행 결과 · 관측값 회신



실행 레인 — Agent (Hermes)

수행만 하고 아무것도 판단하지 않음

외부 입출력

데이터 조회 · 통지 송신 · 파일 작성

루프 유지

정해진 간격으로 회차를 이어 감

이력 저장

모든 대화와 실행 결과를 영속화

↔ 모델 교체가 설정 변경 수준

더 나은 모델이 나오면 판단 레인만 교체



진단이 빠름

루프 · 도구 실패의 원인은 늘 실행 레인 쪽



권한을 한 곳에서 통제

무엇을 할 수 있는지는 Agent 가 정함



◆ KEY INSIGHT

모델 · 채널 · 검색 백엔드는 서로 독립적으로 바뀔 수

다만 모델을 바꾼 직후 첫 요청은 프롬프트 캐시가 초기화돼 비용이 한 번 늘어남

☁ 모델 제공자

교체 가능

- 공식 표 43개 제공자
- 로컬 엔드포인트 (Ollama · vLLM)
- OpenRouter · Nous Portal

⚠ 모델 전환 때 비용

프롬프트 캐시가 초기화돼 첫 요청이 대화 전체를 다시 읽음

💬 채널

교체 가능

- 20개 이상 메시징 플랫폼
- Telegram · Slack · Teams · Email
- 사내 메신저는 플러그인

RUNTIME

Hermes 코어

고정된 실행 엔진

🔍 검색 백엔드

교체 가능

- 기본값 Firecrawl
- SearXNG · Brave · DuckDuckGo
- Tavily · Exa · Parallel · xAI

📄 스킬

쌓이는 계층

- 필요할 때 불러오는 절차 기억
- 모델을 바꿔도 그대로 남음



◆ KEY INSIGHT

인격은 사람이 관리하는 고정값이고 지식은 에이전트가 갱신함 —
학습이 쌓여도 말투와 금지 사항이 바뀌는 사고를 구조로 막음



SOUL.md

어떻게 말할까 — 안정된 인격

작성 주체	사람이 직접 편집
변경 빈도	거의 고정 · 의도적으로만 변경
자동 갱신	대상 아님



skills/*.md

무엇을 전할까 — 경험으로 자라는 지식

작성 주체	에이전트가 스스로 생성
변경 빈도	회차마다 갱신 가능
사람 개입	언제든 열어 수정 · 삭제

SOUL.md 에 담는 네 가지

1

화자 설정

1인칭과 호칭 — 사내 브리핑이면 정중하게, 개인 비서면 친근하게

2

어미와 어조

밝은 구어체인지 담백한 보고체인지 한 줄로 규정

3

금지 사항

과도한 경어 · 전문 용어 남발 · 불필요한 사족 — '하지 마라'가 더 효과적

4

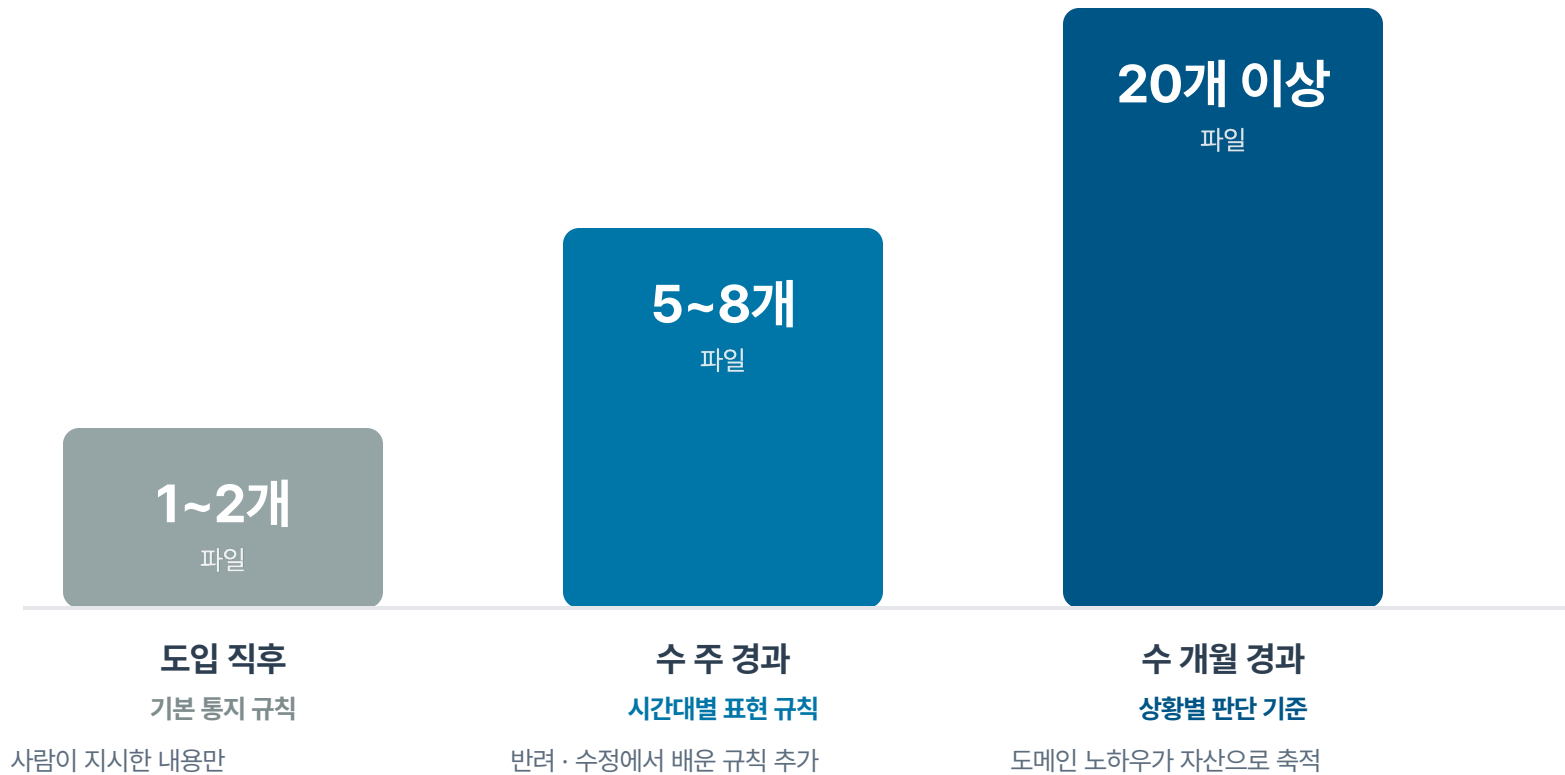
실무 적용

사내 고객 응대 규칙과 커뮤니케이션 가이드를 그대로 옮김



◆ KEY INSIGHT

skills 파일은 반려 · 수정에서 배운 규칙으로 늘어남 —
사람이 읽고 고칠 수 있어 새 담당자에게 넘기는 인수인계 문서 역할도 함



쌓이는 지식의 종류

- 문면 노하우 같은 내용도 더 잘 읽히는 표현
- 회피 규칙 아예 알리지 않는 편이 나은 상황
- 실패 패턴 반려된 판단과 그 이유

사람의 권한

언제든 열어 읽고 직접 수정 · 삭제할 수 있음

매 회차 프롬프트에 실려 판단에 반영됨 — 새 담당자에게 넘기는 인수인계 문서와 같은 성격

PART 3

Hermes Agent 아키텍처

EPISODE 3-2

Gateway 와 메시징 채널 구성

여러 메신저를 한 프로세스로 연결하는 방식과 접근 통제

이 편에서 다루는 내용

- 1 단일 Gateway 의 메신저 연결 구조
- 2 Gateway 전달 보장과 기본 거부 정책
- 3 DM 페어링 인증의 다섯 가지 수치
- 4 외부 호출 없는 사내망 배치 구성



KEY INSIGHT

요청 입구 · 세션 · 예약 실행이 한 프로세스로 모임 —

직원은 새 화면을 배우지 않고 평소 쓰던 메신저로 에이전트에 일을 요청함

메시징 플랫폼 20개 이상

Telegram

Slack

Discord

Microsoft Teams

Google Chat

Email

LINE

Matrix

WhatsApp

Signal

+ 사내 메신저는 플러그인 어댑터로 추가



세션 연결

한 경로에서 시작한 대화를 다른 경로에서 이어 감



Profile 라우팅

봇마다 연결된 Profile 로 요청 전달



크론 실행

예약 작업도 같은 프로세스에서 실행 · 결과 전달



입출력 형식

음성 메모 문자 변환 · 이미지 · 파일 송수신



직원은 새 콘솔 대신 평소 쓰는 대화창에 메시지를 남기면 요청이 시작됨 — 별도 서버 프로세스 없이 하나의 Gateway 가 처리



◆ KEY INSIGHT

허용 목록과 페어링 밖의 사용자는 기본 거부됨 —
채널을 하나 여는 일은 곧 에이전트로 들어오는 접근 경로를 하나 여는 일과 같음

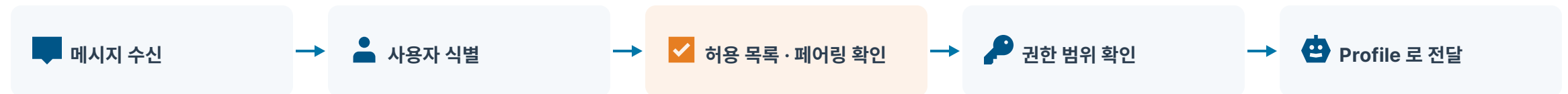
▶ 전달 보장

- 전달 원장 state.db 에 보낼 메시지를 기록
- 최소 1회 전달 실패 시 24시간 안에 최대 3회 재전송
- 서킷 브레이커 어댑터별로 장애를 격리해 전체 중단 방지

🔒 접근 통제

- 기본 거부 허용 목록 · DM 페어링 밖 사용자는 거부
- 권한 구분 관리자와 일반 사용자의 명령 범위 분리
- 범위 구분 DM 과 그룹 대화마다 허용 명령을 따로 지정

메시지 한 건이 들어올 때



운영 정책 예 — 운영 채널에는 검증된 스킬만 노출하고, 시험 채널에서만 실험 스킬을 허용



◆ KEY INSIGHT

코드 길이 · 유효 시간 · 요청 제한 · 대기 상한 · 잠금이 함께 작동함 —
추상적인 인증 설명이 아니라 조직의 접근 통제 기준과 수치로 바로 대조할 수 있음

흐름 사용자가 DM 으로 1회용 코드 요청 → 관리자가 hermes pairing approve <플랫폼> <코드> 로 승인 → 그 계정에 사용 권한 부여

코드 길이

8자

혼동 문자 0 · O · 1 · l 를 뺀 32자
알파벳

유효 시간

1시간

발급 뒤 1시간이 지나면 만료

요청 제한

10분 1회

사용자당 10분에 한 번만 요청

대기 상한

3개

플랫폼당 동시에 대기하는 코드 수

잠금 조건

5회 실패

연속 실패 시 1시간 잠금

✓ 이 수치가 뒷받침하는 것

- 재사용 공격과 무차별 대입을 어렵게 만든 조합 설계
- 조직 기준과 항목별로 직접 대조 가능

? 이 수치가 답하지 않는 것

- 페어링 뒤 세션이 얼마나 오래 유지되는지
- 재인증 주기가 있는지 — 별도 확인 필요



KEY INSIGHT

모델 호출 경로 전체가 사내망 안에 있어 판단 데이터가 밖으로 나가지 않음 —
다만 웹 검색 · 외부 메신저 서버처럼 다른 경로는 따로 확인해야 할 대상임

사내망 경계 — 모델 호출 외부 0건



응답도 사내망 안에서 돌아옴



세션 기록 — 대화 · 산출물 · 판단 근거를 사내 저장소에 보존

경계 밖으로 나갈 수 있는 경로

웹 검색 백엔드

Firecrawl 등 기본값은 외부 서비스

외부 메신저 서버

연동한 플랫폼 서버를 경유

클라우드 모델

설정에 남은 외부 제공자



확인 방법 — 외부 호출 0건을 사업자 약관이 아니라 조직 자신의 로그로 검증

EPISODE 3-3

Profile 과 Kanban 의 역할 분담

다중 에이전트 운영 단위와 작업 인계 방식

이 편에서 다루는 내용

- 1 Profile 의 정의와 격리 범위
- 2 프로파일별 자원과 공유 자원
- 3 Profile 은 샌드박스가 아님
- 4 Kanban 작업 보드의 상태와 속성
- 5 Kanban 의존성 기반 부서 간 인계



Profile 의 정의와 격리 범위

◆ KEY INSIGHT

Profile 하나가 설정 · 메모리 · 스킬 · 크론 · 세션을 따로 가짐 —
역할별로 쓸 수 있는 도구와 데이터를 에이전트 단위로 나눌 수 있음

Profile = 독립된 Hermes 홈 디렉터리

~/.hermes/profiles/<이름>/ (기본 Profile 은 ~/.hermes)

 **config.yaml**
모델 · 도구 · MCP 설정

 **.env**
API 키 · 봇 토큰

 **SOUL.md**
역할 · 말투

 **memories/**
MEMORY.md · USER.md

 **skills/**
절차 스킬

 **cron/**
예약 작업

 **state.db**
세션 기록

 **Gateway**
자체 메신저 봇

만드는 방법

- hermes profile create <이름> — 번들 스킬로 시작
- --clone 으로 기존 Profile 복제 · 생성 뒤에는 원본과 독립
- 채널 설정은 --clone-channels 를 줄 때만 복사

역할별 권한 예

Profile	허용	불가
일반 직원	문서 요약 · 일정 관리	인사 데이터 · 보안 설정
팀장	팀 보고서 · 인사 조회	보안 설정
인사 담당	인사 데이터 · 채용 분석	시스템 관리
IT 관리자	시스템 관리	—



◆ KEY INSIGHT

역할을 가르는 것은 역할 문서 · MCP 연결 · 예약 작업 세 가지임 —
모델 경로는 공유하므로 에이전트를 늘려도 프록시 설정은 바꾸지 않아도 됨

Profile 마다 따로 갖는 자원

자원	정해지는 것
역할 문서 (SOUL.md)	에이전트의 성격 · 임무 · 답변 방식
설정 파일	연결할 MCP 서버와 도구 구성
메신저 봇 토큰	사용자가 대화하는 채널
예약 작업 목록	정해진 시각에 실행할 업무

세션 DB · 메모리	대화 기록과 장기 기억
-------------	--------------

주황 행 = 역할을 가르는 세 가지



주의

같은 컨테이너의 Profile 은 같은 프록시 키로 호출 — 키가 공유되는 구조임

모든 Profile 이 공유하는 자원

- 스킬 디렉터리와 보조 스크립트 문서 읽기 · 요약 같은 공용 기능
- 컨테이너 환경 변수 LiteLLM Proxy 주소와 키

모델 호출 경로



OpenAI 호환 방식 · 에이전트를 늘려도 설정 무변경

Profile 마다 다른 MCP 연결 예

시스템 성능 점검
APM · 위키

영업
CRM · 위키

공공 업무
법령 · 공문서



Profile 은 샌드박스가 아님

3-3 · Profile 과 Kanban 의 역할 분담

KEY INSIGHT

같은 컨테이너의 Profile 은 서로의 파일과 비밀에 닿을 수 있음 —
완전한 격리가 필요한 역할은 컨테이너를 나눠 배치할 대상이 됨

컨테이너 하나

Profile A

설정 · 메모리 · 스킬
세션 · 크론 · 봇

분리됨

Profile B

설정 · 메모리 · 스킬
세션 · 크론 · 봇

분리됨

Profile C

설정 · 메모리 · 스킬
세션 · 크론 · 봇

분리됨



공유되는 영역

파일 시스템 접근 · 컨테이너의 공용 비밀 파일 · 프록시 키

Profile 이 나누는 것과 나누지 않는 것

항목

Profile 사이

설정 · 메모리 · 스킬

나눔

세션 · 크론

나눔

봇 토큰

나눔 — 같은 토큰 중복 사용은 잠금

파일 시스템 접근

나누지 않음

공용 비밀 파일

나누지 않음

완전한 격리가 필요할 때의 선택지



Profile 별 컨테이너 분리

역할마다 컨테이너와 비밀을 따로



쿠버네티스 Secret 분리

Profile 전용 Secret 만 마운트



Docker 터미널 백엔드

명령 실행을 별도 컨테이너에 가둠



◆ KEY INSIGHT

작업 상태 8종과 인계 기록이 SQLite 파일 하나에 남음 —
별도 DB 서버 없이 백업은 파일 복사, 복구는 파일 교체로 끝남

상태 8종

triage	todo	ready	running	blocked	review	done	archived
신규 요청 검토	실행 확정	즉시 실행 가능	에이전트 처리 중	의존성 · 실패 대기	사람 검토	완료	이력 보존

카드 속성

담당 Profile assignee 처리할 Profile 이름 하나	선행 작업 링크 task_links 부모가 모두 done 이면 todo → ready	작업 공간 workspace scratch · worktree · dir:<경로>	tenant tenant 부서 · 프로젝트 격리 단위	댓글 comments 사람과 에이전트의 비동기 기록
--	--	--	--	---

🕒 디스패처

60초마다 ready 작업을 찾아 담당 Profile 워커(hermes -p <담당>)로 실행

⚠️ 실패 처리

연속 실행 실패 2회면 자동 blocked · 1시간 넘는 작업은 시간당 하트비트

📁 운영 이점

SQLite 단일 파일 — 백업은 파일 복사, 복구는 파일 교체



◆ KEY INSIGHT

앞 작업이 **done** 이 되면 다음 작업이 **ready** 로 바뀜 —

인계마다 사람이 보고 고칠 수 있는 기록이 남는 점이 함수 호출식 위임과 다름

인계 흐름 예 — 선행 작업만 선언하면 순서가 자동으로 이어짐



선행 작업이 모두 **done** 이면 다음 작업이 **todo** → **ready** 로 바뀌고, 디스패처가 담당 **Profile** 을 깨움

사람이 개입하는 세 지점

triage 검토

새 요청을 검토하고 담당을 지정

댓글 지시

작업 카드에 추가 지시 · 수정 요청

상태 직접 변경

보드에서 카드를 옮겨 흐름 제어

단계별 자율 비율 예



PART 4

Hermes Agent 핵심 기능

EPISODE 4-1

Skill 시스템의 구조와 작성 방식

절차를 저장하고 필요할 때만 실행하는 방법

이 편에서 다루는 내용

- 1 Skill 의 3단계 로딩
- 2 스킬 수와 실행 비용의 관계
- 3 skill_manage 로 스킬을 직접 작성
- 4 Skill 과 MCP, Hook 의 역할 구분
- 5 업무 에이전트가 남기는 산출물



Skill 의 3단계 로딩

4-1 · Skill 시스템의 구조와 작성 방식

◆ KEY INSIGHT

평소에는 스킬 목록만 싣고 본문과 파일은 필요할 때 싣음 —

그래서 설명 한 줄의 품질이 에이전트가 맞는 스킬을 고르는 정확도를 좌우함

L0 목록

skills_list()

이름 · 설명 · 카테고리

항상 · 약 3k 토큰

L1 본문

skill_view(name)

작업이 설명과 맞을 때 SKILL.md 전체

필요할 때

L2 참고 파일

skill_view(name, path)

실행 중 필요한 scripts · references · assets

절차가 요구할 때

스킬 폴더 구성

my-skill/

SKILL.md

scripts/

references/

assets/

SKILL.md 의 YAML 머리말: name · description · version



설명 한 줄이 중요한 이유

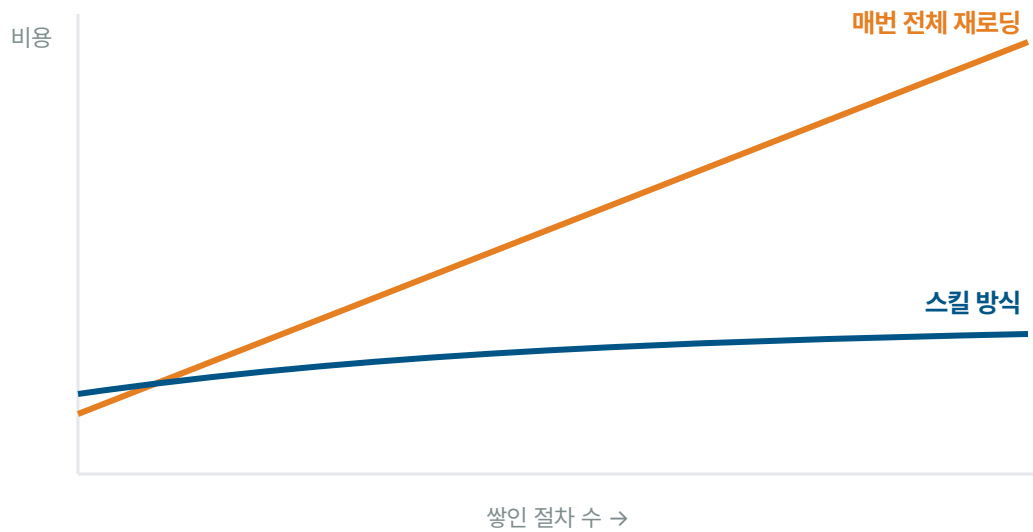
L0 에는 설명만 실리므로, 설명이 모호하면 엉뚱한 스킬을 싣거나 필요한 스킬을 놓침



◆ KEY INSIGHT

필요한 스킬만 펼치므로 절차를 쌓는 일이 실행 부담으로 바뀌지 않음 —
다만 스킬이 많을 때 무관한 스킬을 거르는 정밀도는 공개 자료로 확인되지 않음

스킬 개수에 따른 실행 1회 비용 (개념도)



매번 전체 재로딩

- 실행마다 지침 전체를 처음부터 다시 읽음
- 절차를 하나 더하면 다음 실행 부담도 늘어남

스킬 방식

- 관련 부분만 그때그때 열람
- 한 번 문서로 남기면 다시 구술할 필요 없음

✓ 공개 표준 호환 — agentskills.io

전용 형식이 아니라 공개 명세 — 도구를 바꿔도 형식 차원에서 스킬 자산이 남음

? 확인되지 않은 지점

스킬이 많을 때 무관한 스킬을 거르는 정밀도 · 다른 도구로의 실제 이전 수월성



skill_manage 로 스킬을 직접 작성

4-1 · Skill 시스템의 구조와 작성 방식

◆ KEY INSIGHT

반복할 만한 절차와 사용자 교정이 SKILL.md 로 남음 —
agentskills.io 형식이라 다른 호환 도구에서도 같은 스킬을 읽을 수 있음

🔧 도구가 하는 일

- **create** 새 스킬 생성
- **patch** 부분 수정 (권장)
- **delete** 스킬 삭제
- **write_file** · **remove_file** 참고 파일 관리

💡 생성 계기

- 반복할 만한 다단계 작업
- 발견한 오류의 복구 경로
- 사용자가 교정해 준 접근 방식
- system prompt 의 기록 지시

📖 권장 구성 4절

- **When to Use** 언제 쓰는지
- **Procedure** 절차
- **Pitfalls** 함정
- **Verification** 검증 방법

흐름 예

저장 위치: ~/.hermes/skills/ · 프로젝트 .hermes/skills/

사용자가 절차를 교정



에이전트가 skill_manage 호출



SKILL.md 생성 · 저장



다음 실행부터 LO 목록에 노출



◆ KEY INSIGHT

판단이 필요한 절차는 Skill, 판단 없이 매번 돌 일은 Hook 이 맡음 —
도구를 연결하는 것만으로는 그 도구를 쓰는 방법이 전달되지 않음



MCP

Model Context Protocol

역할

외부 시스템을 도구로 연결하는 규약

만들 일

외부 시스템용 도구 — 위키 · CRM · DB

도구를 줌



Skill

절차 문서와 스크립트

역할

그 도구를 쓰는 방법을 알려 줌

만들 일

판단이 필요한 절차 — 예: 릴리스 점검 목록

쓰는 법을 알려 줌



Hook

정해진 시점의 스크립트

역할

판단 없이 매번 실행

만들 일

판단이 필요 없는 일 — 예: 저장 시 포맷 · 위험 명령 거부

매번 강제함

표준화 현황 — Skill · AGENTS.md · MCP 는 개방 표준, Hook 과 서브에이전트 정의 형식은 공통 표준 없음



◆ KEY INSIGHT

대화로 시킨 일이 스킬 · 예약 작업 · 작업 카드로 영속됨 —
검증 없이 쌓이면 바이브코딩과 같은 검증 부재 · 권한 과다 문제가 그대로 생김

항목	coding agent	업무 에이전트
구조	도구를 루프로 돌리는 모델	도구를 루프로 돌리는 모델
절차 저장	Agent Skills 형식 (SKILL.md 폴더)	같은 Agent Skills 형식
대표 도구	Claude Code · Cursor	Hermes Agent · OpenClaw
사람이 하는 일	코드를 읽지 않고 결과만 보고 다음 지시	스크립트를 읽지 않고 결과만 보고 다음 지시
생기는 문제	검증 수단 부재 · 과도한 권한	같은 형태로 나타남

대화가 남기는 산출물



스킬

SKILL.md — 절차 문서



예약 작업

Cron — 정해진 시각 실행



작업 카드

Kanban — 위임 큐



대응

에이전트가 받는 피드백과 넘지 못하는 경계를 설계 — 산출물을 사람이 검토하는 절차를 함께 둬

PART 4

Hermes Agent 핵심 기능

EPISODE 4-2

Memory 와 학습 루프

Hermes 가 기억하는 범위와 스스로 고치는 방식

이 편에서 다루는 내용

- 1 Hermes Memory 의 구성
- 2 메모리와 스킬의 저장 대상 구분
- 3 다섯 단계 자기학습 루프
- 4 백그라운드 리뷰와 Curator 주기
- 5 Markdown 학습 결과의 감사 · 복구 이점



◆ KEY INSIGHT

항상 실는 메모리는 두 파일을 합쳐 약 3,600자로 작음 —
나머지 과거는 세션 검색과 외부 메모리 제공자가 맡아 필요할 때만 꺼냄

MEMORY.md

2,200자

약 800 토큰

환경 정보 · 작업 사실

항상 실림

USER.md

1,375자

약 500 토큰

사용자 정보와 선호

항상 실림

세션 검색

상한 없음

SQLite FTS5 · 약 20ms

모든 CLI · 메신저 세션 기록

필요할 때 검색

외부 제공자

8종

Honcho · Mem0 · Hindsight 등

벡터 기반 장기 기억 연동

선택 연동

세션 시작 때 프롬프트에 고정 스냅샷으로 실림

프롬프트 밖 — 에이전트가 필요할 때 조회

⚠ 상한을 넘으면

자동 압축이 없어 오류가 돌아옴 — 에이전트가 항목을 정리하거나 지운 뒤 다시 저장하며, 80% 를 넘으면 정리를 권장

KEY INSIGHT

메모리에는 사실을, 스킬에는 절차를 둬 —
작은 메모리에 절차까지 넣으면 상한에 걸려 정작 필요한 사실이 밀려남

메모리 (MEMORY.md · USER.md)

무엇이 사실인가

- **환경 정보** 서버 이름 · 경로 · 사용 도구
- **사용자 선호** 보고 형식 · 호칭 · 언어
- **프로젝트 사실** 담당자 · 기한 · 결정 사항

예) 리포트는 표 형식을 선호함

스킬 (SKILL.md)

어떻게 하는가

- 다단계 작업 절차
- 오류 복구 경로와 함정
- 결과 검증 방법

예) 월간 손익 리포트 작성 절차 7단계

반영 시점과 도구

🕒 세션 시작 때 고정

메모리는 세션 시작 시점의 스냅샷이 실림 — 세션 중 고친 내용은 다음 세션부터 반영

✅ memory 도구

add · patch · remove 세 동작 · 저장 전에 주입 · 유출 패턴을 검사



◆ KEY INSIGHT

사람의 승인 · 편집 · 반려가 다음 회차 판단 기준으로 반영됨 —
별도 학습 데이터를 만들지 않아도 운영 중 피드백만으로 판단이 개선됨



1



실행

판단 결과대로 통지 · 작업 처리

2



결과 수집

승인 · 편집 · 반려 확인

3



반성

원문과 수정문 차이에서 규칙 추출

4



스킬 출력

규칙을 Markdown 스킬로 작성

5



반영

저장 후 다음 프롬프트에 포함



KEY INSIGHT

약 10턴마다 리뷰가 스킬을 만들고 Curator 가 7일마다 쓰지 않는 스킬을 정리함 —
자동 삭제 없이 보관만 하므로 잘못 정리된 스킬도 언제든 되돌릴 수 있음

백그라운드 리뷰

약 10 에이전트 턴 주기

- 턴이 끝난 뒤 리뷰 포크가 회고
- 재사용할 절차를 스킬로 생성 · 우산 스킬 작성
- 사람이 지켜보지 않는 시간에도 절차를 문서화

Curator

7일(168시간) 주기 · 유틸 2시간 이상

- 에이전트가 만든 스킬의 사용 기록 추적
- 쓰지 않는 스킬을 단계적으로 정리
- LLM 통합 리뷰는 기본 꺼짐 (1회 50~100 호출)

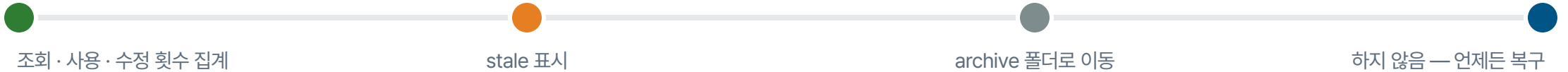
스킬 노후 기준

사용 중

14일 미사용

30일 미사용

자동 삭제



보호 — pin 한 스킬과 크론이 참조하는 스킬은 자동 정리에서 제외 · 허브에서 설치한 스킬은 관리 대상 아님



◆ KEY INSIGHT

학습 결과가 문서로 남아 잘못 배운 규칙만 골라 지울 수 있음 —
그대로 감사 제출 자료와 새 담당자 인수인계 문서로 쓸 수 있음

일반적인 학습 — 가중치 갱신

무엇을 배웠는지 알 수 없음

- 학습 결과가 수치 파라미터로만 존재
- 특정 규칙만 골라 되돌리기 어려움
- 왜 그렇게 판단했는지 설명하기 어려움

문제 발생 시 — 전체 재학습 외 선택지가 좁음

Hermes — Markdown 파일 갱신

열어서 읽고 고치고 지울 수 있음

- 학습 결과가 사람이 읽는 문장으로 존재
- 잘못 배운 규칙만 골라 즉시 삭제
- 판단 근거를 문서로 그대로 제시

문제 발생 시 — 그 줄을 지우면 다음 회차부터 원복

운영 · 규제 관점의 이점

감사 가능성

언제 어떤 규칙이 추가됐는지 파일 이력으로 추적 · 내부 감사와 외부 심사에 제출

사고 대응 속도

잘못 학습된 규칙만 제거 — 재학습 · 롤백 배포 없이 분 단위 복구

인수인계 자산

축적된 스킬 파일이 곧 도메인 매뉴얼 — 에이전트를 건너내도 지식은 남음

PART 4

Hermes Agent 핵심 기능

EPISODE 4-3

Cron 과 무인 실행

사람이 없는 시간에 일을 맡기는 방법과 한도

이 편에서 다루는 내용

- 1 Cron 예약 작업의 생성과 실행
- 2 Cron 과 Kanban 의 작업 투입 방식 차이
- 3 무인 실행의 사람 개입 지점
- 4 서브에이전트 위임 방식



◆ KEY INSIGHT

Cron 작업은 대화 맥락이 없는 새 세션에서 실행됨 —

대화에서 합의한 맥락은 작업 프롬프트나 연결한 스킬에 적혀 있어야만 전달됨

만드는 방법



"매일 오전 9시에 AI 소식을 확인해 메신저로 요약을 보내줘" — 대화 한 문장으로 생성

일정 표기

in 30m

every 2h

weekdays at 9am

0 9 * * 1-5

CLI · 명령으로도 생성 가능 · ISO 타임스탬프 1회 실행 지원

실행 방식

- **새 세션** 대화 플랫폼이 붙지 않은 빈 세션에서 실행
- **확인 주기** Gateway 가 60초마다 실행할 작업을 확인
- **기록** 작업별 출력 파일 · 실행 원장에 결과 저장
- **전달** 원래 대화 · 메신저 채널 · 로컬 파일 중 지정

한도와 규칙

3,600초

스크립트 실행 제한 시간

3회

연속 실패 시 알림

5 · 15 · 30분

모델 불통 시 재시도 간격

금지

크론 세션 안에서 새 크론 생성



no_agent 모드 — LLM 없이 스크립트만 실행 · 출력이 비면 메시지를 보내지 않아 토큰을 쓰지 않음



◆ KEY INSIGHT

시각 기준 반복은 Cron, 담당 간 인계는 Kanban 이 맞음 —
다른 Profile 에 일을 넘기는 기본 경로는 Gateway 호출이 아니라 Kanban 작업 카드임

구분	Cron	Kanban
기준	시각 · 간격	작업 · 담당
저장 위치	Profile 마다 따로 (cron/jobs.json)	모든 Profile 이 공유 (kanban.db)
시작 조건	정해진 시각이 되면	선행 작업이 모두 done 이 되면
결과	정해진 채널 · 파일로 전달	작업 카드 상태 · 댓글로 기록
맞는 일	매일 리포트 · 주간 집계 · 정기 점검	요청 → 검토 → 발행 같은 다단계 인계

⚠ Kanban 작업의 함정

목표(goal)를 주지 않으면 한 턴만 실행 · 디스패처가 꺼져 있으면 ready 에서 멈춤

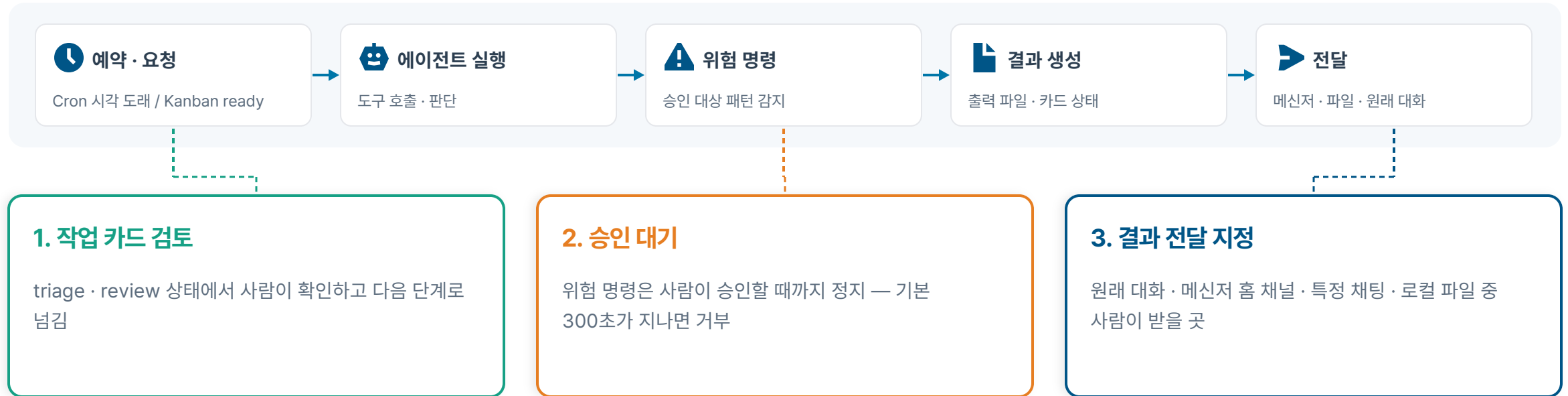
➡ 선택 기준

언제 할지가 정해진 일 → Cron · 누가 이어받을지가 정해진 일 → Kanban



◆ KEY INSIGHT

승인 대기 · 결과 전달 · 작업 카드 검토 세 곳에서 사람이 개입함 —
실패 알림은 그 작업의 전달 대상에게만 가므로 전달 대상이 비면 실패가 묻힘



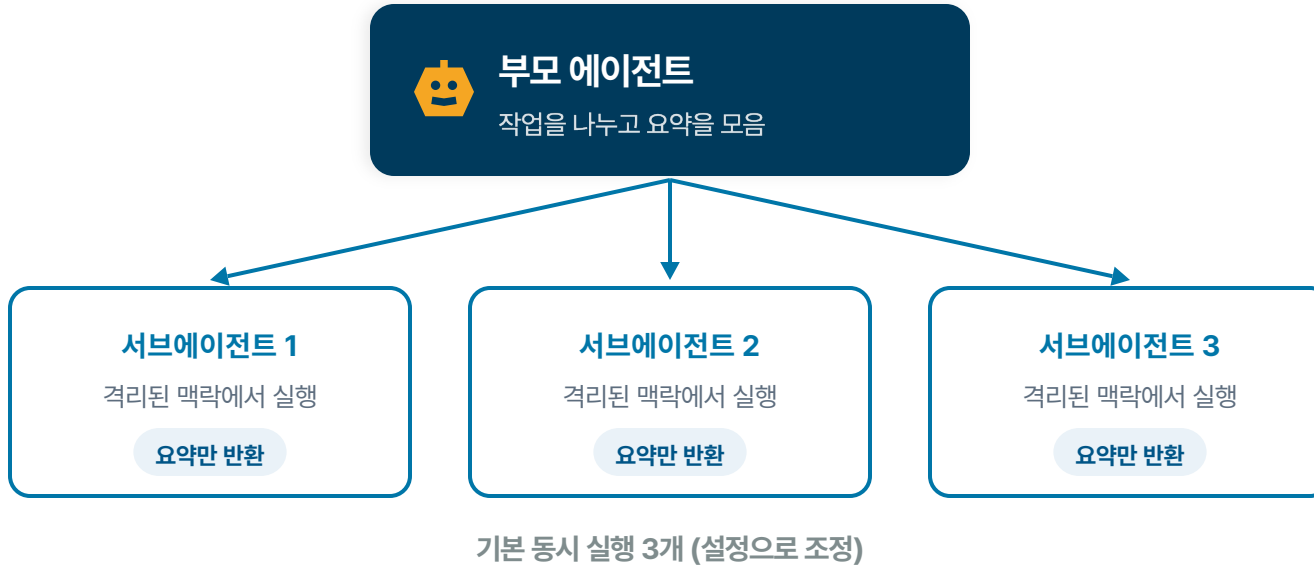
🚩 놓치기 쉬운 함정 — 실패 알림의 경로

실패 알림은 그 작업의 전달 대상으로만 감 — 전달 대상이 local(파일)인 작업은 실패해도 사람에게 알림이 가지 않음. 알림이 갔다는 기록이 사람이 받았다는 뜻은 아님



KEY INSIGHT

서브에이전트는 격리된 맥락에서 일하고 요약만 돌려줌 —
메모리 · 크론 · 메시지 발송 도구가 막혀 있어 부모 에이전트의 권한을 넘지 못함



서브에이전트에서 막힌 도구

- 하위 위임 `delegate_task` — 손자 단계로 다시 위임 불가
- 사용자 질문 `clarify` — 사람에게 직접 묻지 못함
- 메모리 `memory` — 부모의 기억을 바꾸지 못함
- 메시지 발송 `send_message` — 외부로 직접 보내지 못함
- 크론 `cronjob` — 예약 작업을 만들지 못함



맥락

부모의 SOUL.md · 메모리를 신지 않음 · 부모 Profile 에 고정



쓰임

결가지 조사 · 대량 파일 처리처럼 출력이 대화를 채우는 작업



이점

본 대화의 맥락을 깨끗하게 유지하고 권한도 좁게 유지

PART 4

Hermes Agent 핵심 기능

EPISODE 4-4

MCP 와 LiteLLM 기반 외부 연동

도구 · 모델 표준과 설치 요건

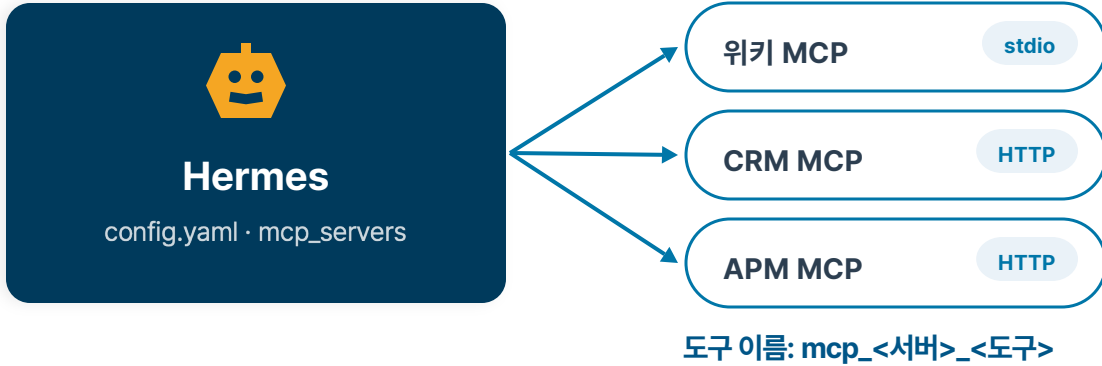
이 편에서 다루는 내용

- 1 MCP 서버 연동 방식
- 2 모델은 LiteLLM, 도구는 MCP 로 분리
- 3 웹 검색 백엔드 자동 감지
- 4 설치 요건과 서버 규격 산정



KEY INSIGHT

등록한 MCP 서버의 도구가 `mcp_<서버>_<도구>` 이름으로 에이전트에 붙음 —
포함 · 제외 목록으로 좁혀야 에이전트가 쓸 수 있는 도구가 필요한 만큼으로 줄어듦



자격 증명 필터

stdio 서버는 셸 환경 변수를 물려받지 않음 — 명시한 변수와 안전 변수만 전달

PATH

HOME

USER

LANG

LC_ALL

TERM

SHELL

TMPDIR

항목

내용

전송 방식

stdio · HTTP(Streamable) · 레거시 SSE

인증

OAuth 2.1 + PKCE · mTLS

노출 통제

tools.include / tools.exclude (glob)

동시 탐색

서버 최대 4개 · 탐색 제한 시간 설정

Hermes 를 MCP 서버로

공식 플러그인 mcp-serve 설치

⚠ 운영 팁

- 쓰기 도구는 기본으로 막고 필요한 것만 허용
- 오류 메시지의 토큰 · 키는 [REDACTED] 로 가려짐
- 서브에이전트는 부모의 MCP 도구를 물려받음



KEY INSIGHT

Hermes 는 두 표준의 클라이언트로만 동작함 —
모델이나 도구를 바꿀 때 Hermes 쪽 코드와 설정을 거의 고치지 않아도 됨



Hermes Agent

두 표준의 클라이언트



LiteLLM — 모델 게이트웨이

OpenAI 호환 주소 하나



MCP — 도구 표준

사내 시스템을 한 번 노출

Amazon
Bedrock

Azure OpenAI

Vertex AI

로컬 vLLM ·
Ollama

위키

CRM

GitHub

사내 DB

LiteLLM Proxy 가 맡는 통제

- **비용** 키 · 사용자 · 팀 단위 예산과 호출 한도
- **권한** Virtual Key 발급 · 삭제 · 모델 접근 제한
- **안정성** API 장애 시 다른 모델로 폴백
- **확장** 관측 도구 · 가드레일 연동

통로가 하나가 되면 생기는 것

교체 비용 감소

모델 · 도구를 바꿔도 Hermes 설정은 주소 하나

통제 지점 단일화

비용 · 권한 통제를 Proxy 한 곳에 걸 수 있음

중립 표준

MCP 는 Linux Foundation 산하 중립 거버넌스



◆ KEY INSIGHT

다른 용도로 등록된 키 하나가 검색 요청의 목적지를 바꿀 수 있음 —
설정한 백엔드와 실제로 켜진 백엔드가 같은지가 도입 점검 항목이 됨

검색 · 추출 백엔드

Firecrawl

기본값

SearXNG

Brave

DuckDuckGo

Tavily

Exa

Parallel

xAI

SearXNG 는 자체 호스팅 대표 선택지

설정하지 않으면 API 키가 백엔드를 대신 고름

백엔드 미지정

등록된 API 키 확인

사용 가능한 서비스 자동 선택

검색어 · 추출 결과가 그 서비스로

조직이 고를 수 있는 세 갈래

외부 서비스 사용

기본값 유지 — 검색어와 추출 결과가 외부 서비스를 경유

자체 인프라로

SearXNG 등 자체 호스팅 — 경로가 조직 인프라 안에 머물

기능 끄기

외부 검색 자체를 막아야 하는 조직은 별도 비활성 구성



◆ KEY INSIGHT

공식 문서는 런타임 조건만 밝히고 CPU · 메모리 최소치는 적지 않음 —
실제 부하로 시험 설치해 보기 전에는 서버 구매 사양을 확정할 수 없음

✓ 공식 문서가 밝힌 조건

- 런타임 Python 3.14 기본 · 지원 3.11 ~ 3.14
- 설치기 Python · uv · Node 를 직접 내려받음
- OS Linux · macOS · WSL2 · Windows · Android(Termux)
- 데스크톱 앱 macOS 는 Apple Silicon 전용
- 모델 조건 컨텍스트 64K 이상

? 공식 문서에 없는 값

- 최소 CPU 코어 수
- 최소 메모리 용량
- 최소 디스크 용량
- 동시 사용자 수별 규격

→ 작업량에 맞는 사양은 시험 설치로 직접 측정

참고 수치 — 공식 기본값

CPU 1

컨테이너 터미널 기본

5,120MB

컨테이너 메모리 기본

180초

명령 기본 제한 시간

8GB+

내장 llama.cpp 소형 모델 GPU

EPISODE 5-1

로컬 LLM 이 필요한 업무와 이유

데이터 주권 · 비용 · 상시 실행 관점

이 편에서 다루는 내용

- 1 클라우드 LLM 의 두 가지 제약
- 2 자체 호스팅이 통제하는 범위
- 3 로컬 LLM 결합의 네 가지 이점
- 4 LLM 단가 하락과 과업별 모델 배치
- 5 로컬 LLM 에 대한 다섯 가지 질문



◆ KEY INSIGHT

기밀 데이터와 상한 없는 과금이 클라우드 사용을 막음 —

로컬 LLM 은 클라우드를 못 쓰는 업무에 한정해 들일 때 비용 대비 효과가 가장 큼



기밀 데이터

외부 전송이 허용되지 않는 데이터

- 클라우드 API 호출 자체가 불가
- 망분리 · 개인정보 · 영업 비밀이 걸린 업무
- 약관만으로는 내부 규정을 통과하지 못함



종량 과금

비용 상한을 알 수 없음

- 예산 통제 수단이 없으면 사용 제한으로 이어짐
- 상주 자동화일수록 호출 수가 계속 늘어남
- 개인별 자유로운 사용을 허용하기 어려움



로컬 LLM 이 해법이 되는 이유

데이터가 사내를 벗어나지 않음

모델 호출 경로 전체가 내부망

사용량이 늘어도 추가 과금 없음

하드웨어 상각 · 전력 고정비

엔지니어가 제한 없이 사용

상주 자동화도 비용 걱정 없음



◆ KEY INSIGHT

자체 호스팅은 코어가 도는 위치만 사내로 옮김 —
검색 · 모델 · 사용 정보 · 메신저 경로를 하나씩 확인해야 사내 완결이 성립함

데이터가 나가는 경로	기본값	전환 가능 여부	판단
검색 · 추출 백엔드	Firecrawl (API 키 기준 자동 감지)	자체 호스팅 전환 가능	사내로 돌릴 수 있음
모델 제공자	선택 시점에 지정 (Anthropic · OpenRouter · Nous 등)	전환 가능 · 전환 시 캐시 초기화	로컬 지정 시 사내
사용 정보 수집	신규 계측은 동의 전까지 비활성	정책과 현황을 따로 확인	목적지 확인 필요
메신저 Gateway	연동한 플랫폼 서버를 경유	채널 선택에 따라 다름	플랫폼 약관 대조

🗨 자체 호스팅이 가두는 것은 코어가 도는 위치이며, 코어가 만드는 네 갈래 호출까지 자동으로 사내에 가두지는 않음



로그로 검증

외부 호출 0건을 사업자 약관이 아니라 조직의 로그로 직접 확인



추적 가능성

어느 모델에 무엇이 흘렀는지 경로별로 기록



모델 교체 자유

LiteLLM 뒤에서 모델을 바꿔도 경로는 그대로



◆ KEY INSIGHT

비용과 상시성이 서로를 강화함 —

예약 작업처럼 상주 자동화가 많은 조직일수록 로컬 결합의 경제성이 빠르게 커짐



비용

01

토큰 과금 대신 하드웨어 상각과 전력만 남음 — 고정비 중심의 예측 가능한 비용 구조

유의 — 초기 하드웨어 · 전력 고정비 발생



데이터 주권

02

사내 데이터가 외부 SaaS 로 나가지 않고 온프레미스 경계 안에 머물

유의 — 운영 책임이 조직에 있음



상시 실행 (24/7)

03

Rate Limit · 과금 걱정 없이 쉬지 않는 자동화 — 일간 리포트 · 지표 크론에 적합

유의 — 상주 프로세스 모니터링 필요



결정적 처리 결합

04

정해진 렌더러 + 좁은 LLM 역할로 낮은 모델 성능을 보완 — LLM 이 틀려도 시스템은 틀리지 않게

유의 — 결정적 처리와 병행 설계 전제

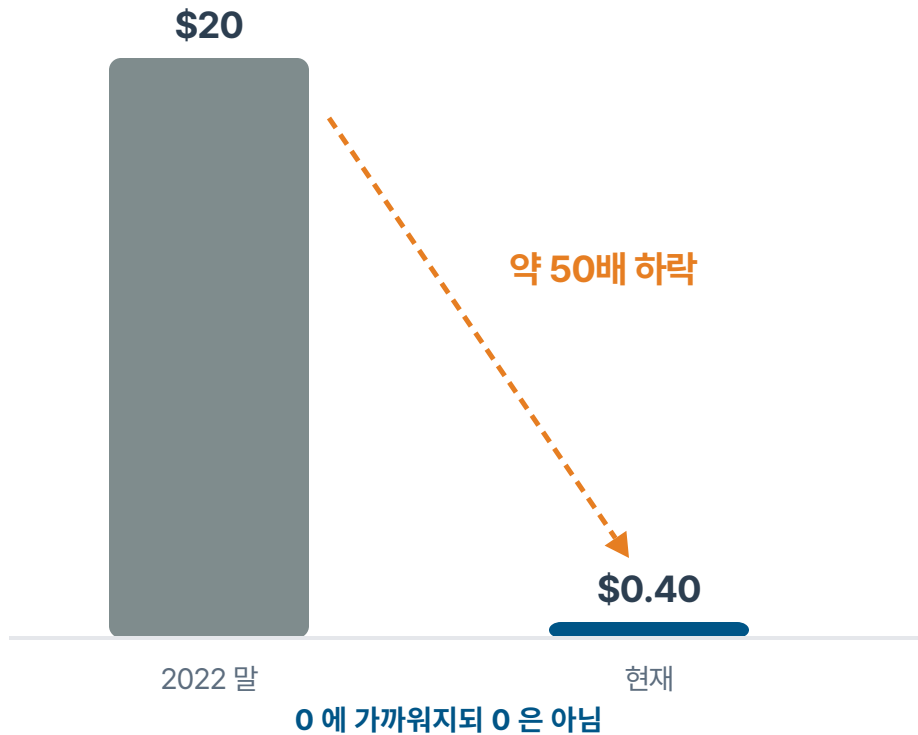


◆ KEY INSIGHT

GPT-4 급 성능의 토큰 단가가 약 50배 떨어짐 —

반복 정형 업무를 소형 모델로 옮길수록 비용 이점이 커지고, 단가는 0에 가까워지되 0은 아님

GPT-4 급 성능 · 100만 토큰당 단가



과업 유형별 모델 배치

개방 · 창의 소수 과업

복잡 설계 · 비정형 추론 · 창의 생성

프런티어 모델 API

저빈도 · 고난도 — 성능이 품질을 좌우

반복 · 정형 다수 과업

문서 분류 · 정형 리포트 · 스크리닝

로컬 소형 모델

상시성 · 비용 · 데이터 주권

상시 백오피스

일간 리포트 · 지표 크론

로컬 모델 + 결정적 렌더러

과금 없음 · 재현성



로컬 LLM 에 대한 다섯 가지 질문

5-1 · 로컬 LLM 이 필요한 업무와 이유

◆ KEY INSIGHT

다섯 질문 대부분이 설정과 모델 크기 문제임 —

주요 도구가 OpenAI 호환 API 를 제공해 애플리케이션은 주소만 바꾸면 전환됨

Q1

애플리케이션을 다시 만들어야 하나

Ollama · vLLM · LM Studio 모두 OpenAI 호환 API — 주소만 바꾸면 전환

Ollama · vLLM · LM Studio 문서

Q2

로컬 LLM 은 성능이 낮은가

모델 크기에 좌우 — 업무 과제로 직접 비교 평가 필요

공개 비교 수치 없음

Q3

매우 비싼 GPU 가 필요한가

4bit 기준 10억 파라미터당 약 0.6~0.7GB — 20B 급은 16GB 안팎

Ollama 모델 파일 크기

Q4

여러 사람이 동시에 쓸 수 있나

Ollama 는 동시 처리 설정 필요 · vLLM 은 연속 배칭 제공

Ollama FAQ · vLLM 문서

Q5

쉽게 실행할 수 있나

설치와 실행 명령이 한두 줄

공개 샘플 저장소

EPISODE 5-2

로컬 모델과 하드웨어 선택

모델 후보 · GPU 메모리 산정 · Hermes 연결 조건

이 편에서 다루는 내용

- 1 로컬 LLM 후보 세 계열 비교
- 2 Gemma 4 26B MoE 사양
- 3 모델 크기별 GPU 메모리 어림
- 4 Hermes 와 로컬 모델 연결 조건



로컬 LLM 후보 세 계열 비교

5-2 · 로컬 모델과 하드웨어 선택

◆ KEY INSIGHT

모든 업무에 맞는 단일 모델은 없음 —

후보를 LiteLLM 에 함께 등록해 업무 유형별로 비교한 뒤 정하는 방식이 실패 비용이 낮음

Gemma 4

Google

구성

E2B · E4B · 12B · 26B A4B(MoE) · 31B Dense

컨텍스트

128K (E2B · E4B) · 256K (12B 이상)

특징

텍스트 + 이미지 · 함수 호출 · 사고 모드

라이선스

Apache 2.0

Qwen3

Alibaba

구성

30B-A3B(MoE, 활성 3.3B) · 32B · 14B · 8B

컨텍스트

128K · 2507 판은 262K

특징

다국어 · 4B 이하는 32K 라 Hermes 조건 미달

라이선스

Apache 2.0

gpt-oss

OpenAI

구성

20b (21B · 활성 3.6B) · 120b (117B · 활성 5.1B)

컨텍스트

128K

특징

20b 는 16GB · 120b 는 80GB GPU 1장

라이선스

Apache 2.0

한글 한국어 성능 — 세 계열 모두 공식 한국어 벤치마크가 없음 → 업무 과제 100건 정도로 자체 평가 후 결정



Gemma 4 26B MoE 사양

5-2 · 로컬 모델과 하드웨어 선택

◆ KEY INSIGHT

매 추론에 전체 파라미터의 약 15% 만 쓰는 MoE 구조임 —

소비자 GPU 한 대로 상주 에이전트를 돌릴 수 있어 하드웨어 진입 장벽이 낮아짐

파라미터

3.8B / 25.2B

추론 시 활성화 / 총 파라미터

컨텍스트

256K

긴 문맥을 한 번에 처리

라이선스

Apache 2.0

사내 배치와 개조에 제약 없음

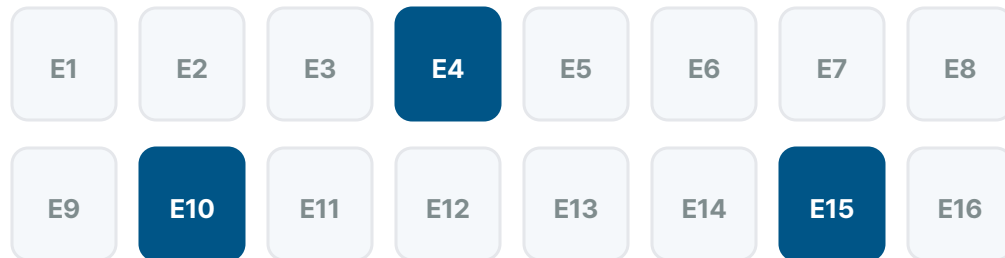
언어

140개 이상

한국어 포함 다국어

MoE (Mixture of Experts)

입력마다 필요한 전문가 블록만 켜짐



계산량은 소형 모델 수준 · 지식 용량은 대형 모델 수준

에이전트로 쓰기 위한 특성

- 네이티브 함수 호출 · 구조화 출력 도구를 정확한 형식으로 호출
- 사고 모드 전환 쉬운 판단은 빠르게, 어려운 판단은 단계 추론
- 텍스트 + 이미지 입력 시각 토큰량도 조정 가능
- 소비자 GPU · 워크스테이션 한 대로 상주 가능



KEY INSIGHT

가중치 크기만으로 VRAM 을 잡으면 부족함 —

컨텍스트 길이와 동시 처리 수만큼 KV 캐시가 더해지므로 가중치의 1.2배 이상을 하한으로 잡음

4bit 양자화 모델 파일 크기 (GB)

qwen3:14b

14B

9.3GB

gpt-oss:20b

20B

14GB

deepseek-r1:32b

32B

20GB

llama3.3:70b

70B

43GB

gpt-oss:120b

120B

65GB

qwen3:235b

235B

142GB

VRAM 별 대략 범위

12GB

12 ~ 14B 모델

24GB

20 ~ 32B 모델

48GB

70B 모델

그 이상

여러 장 · 통합 메모리 장비

어림 규칙

- 4bit 기준 10억 파라미터당 약 0.6~0.7GB
- KV 캐시는 컨텍스트 · 동시 처리 수만큼 추가
- 가중치 크기의 1.2배 정도를 VRAM 하한으로



KEY INSIGHT

64K 미만 컨텍스트는 Hermes 시작 단계에서 거부됨 —

Ollama 기본값과 Qwen3 4B 이하 모델은 설정을 바꾸지 않으면 그대로 쓸 수 없음

연결 전 확인할 조건



최소 컨텍스트 64K (65,536 토큰)

미만이면 시작 단계에서 거부



Ollama 컨텍스트 설정

기본값이 작아 명시적으로 늘려야 함



Qwen3 4B 이하 제외

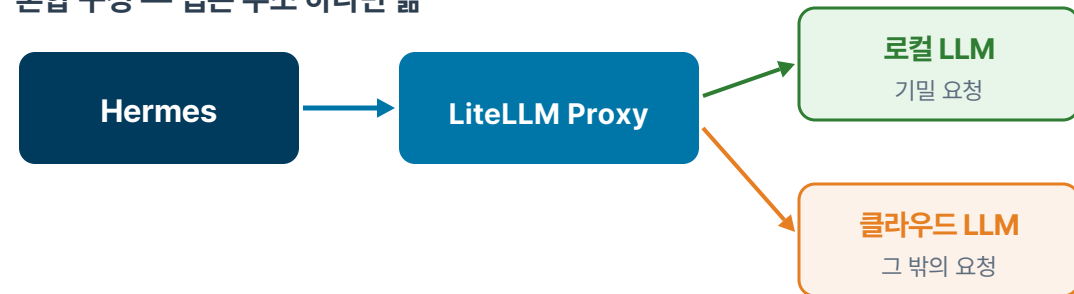
컨텍스트 32K 라 조건 미달



내장 llama.cpp 관리

GPU 8GB+ 소형 · 16GB+ 27~35B · 4bit 미만 미제공 · 유휴 15분 뒤 언로드

혼합 구성 — 앱은 주소 하나만 앎



config.yaml (발췌)

model_name: local-llm model: ollama_chat/<모델>

api_base: http://host.docker.internal:11434

연결할 때 주의할 점

- Docker 로 띄운 Proxy 에서 localhost 는 컨테이너 자신 — 호스트의 Ollama 는 host.docker.internal 로 지정
- Tool Calling 미지원 모델은 JSON 모드로 대체되므로 모델별 supports_function_calling 을 명시 · 로컬 모델은 비용을 직접 입력

EPISODE 6-1

오픈소스 에이전트와 Hermes 비교

OpenClaw · Paperclip 등과의 역할 차이

이 편에서 다루는 내용

- 1 OpenClaw 와 Hermes 의 역할 차이
- 2 Paperclip 과 Hermes 의 계층 차이
- 3 오픈소스 에이전트 5종 비교
- 4 런타임별 절차 단위와 예약 단위



◆ KEY INSIGHT

OpenClaw 는 채널과 도구를 넓게 잇고 Hermes 는 특정 업무의 판단을 오래 키움 —
성장 방식이 달라 대체재보다는 함께 배치하는 보완재로 쓰이는 경우가 많음

OpenClaw

393,513

채널 · 도구를 넓게 잇는 개인 비서

GitHub 스타

Hermes Agent

258,989

특정 업무를 오래 키우는 에이전트

GitHub 스타

한 줄 정의

메시징 게이트웨이 중심 개인 비서

자기 개선형 에이전트 기반

강점

브라우저 · 파일 · API 조작 · 로컬 실행

기억 · 스킬 생성 · 자기 개선 루프

성장 방식

공용 스킬을 가져다 씀 (스킬 허브)

경험에서 스킬을 만들어 냄

운영 주체

OpenClaw 재단 (비영리)

Nous Research

적합 용도

일상 실무 대행 — 매일 부르는 일

특정 영역 연구 · 운영 — 오래 키우는 일



◆ KEY INSIGHT

Paperclip 은 에이전트를 만들지 않고 조직도 · 예산 · 책임으로 묶음 —
Hermes 를 직원으로 등록하는 어댑터를 공식 지원해 두 제품을 함께 쓸 수 있음

Paperclip — 오케스트레이션 계층

MIT · 스타 109,089

"에이전트를 만들지 않고, 기존 에이전트를 채용해 조직으로 운영"

조직도 · 보고선

예산 80% 경고 · 100% 자동 정지

하트비트 · 책임 추적

Hermes

OpenClaw

Claude Code

Codex 등

에이전트 계층 — 각자 일하는 "직원"

언제 무엇을 쓰나

Paperclip

이미 여러 에이전트를 운영 중이고 예산 · 책임 배분 관리가 필요할 때

Hermes

한 업무의 경험을 쌓아 성능을 높이는 것이 목표일 때

함께

Paperclip 조직도 안에 Hermes 를 한 직원으로 등록

구분	Paperclip	Hermes
역할	여러 에이전트 조율 · 관리	개별 에이전트 실행 · 학습
자체 추론 엔진	없음 — 등록된 에이전트가 수행	있음 — 모델 + 학습 루프
지원 어댑터	OpenClaw · Claude Code · Codex · Cursor · Gemini CLI · Hermes 등	—



◆ KEY INSIGHT

학습 루프와 지속 기억을 함께 내장한 점이 Hermes 의 차이임 —
채널 폭과 다중 에이전트 조율은 다른 제품이 앞서므로 용도에 따라 고를 대상임

항목	Paperclip	OpenClaw	MS Agent Framework	CrewAI	Hermes Agent
라이선스	MIT	MIT	MS 약관	MIT	MIT
메시징 채널	내장 없음	넓음 (게이트웨이 중심)	REST · MCP	MCP	20개 이상 플랫폼
학습 루프	없음 (집계만)	사용 이력 로그	Evals 패턴	작업 피드백	백그라운드 리뷰 + Curator
지속 기억	구현 필요	파일 기반	구현 필요	구현 필요	메모리 + 세션 검색
조직화 방식	조직도 · 예산	채팅 워크스페이스	멀티에이전트 패턴	역할 기반 크루	Profile 분화 · Kanban
상업 모델	순수 OSS	스킬 마켓	Azure 연계	구독 플랜	순수 OSS

읽는 법

Paperclip · CrewAI 의 조직화는 '여러 에이전트를 조율' 하는 방식이고, Hermes 는 '한 에이전트가 여러 Profile 로 분화' 하는 방식이라 접근 자체가 다름 — 점수보다 용도로 비교



◆ KEY INSIGHT

벤더 실행형은 한도 · 과금 · 데이터 경계가 벤더 정책에 묶임 —
자가 호스팅은 그 제약이 없는 대신 실행 환경을 켜 두는 운영 책임을 조직이 짐

런타임	절차 단위	예약 · 트리거 단위	실행 위치
Hermes Agent	SKILL.md (에이전트 작성)	cron · kanban	자가 호스팅
OpenClaw	SKILL.md · ClawHub	heartbeat · cron	로컬 자가 호스팅
Claude Code	Skills · plugins	routines (schedule · API · webhook)	Anthropic 인프라
Claude Cowork	Skills · plugins	scheduled task	사용자 PC (켜져 있어야 실행)
ChatGPT	Skills · Workspace Agents	Scheduled tasks	OpenAI 클라우드
Copilot Studio	agent · agent flow	manual · 이벤트 · schedule	Microsoft 클라우드
Gemini Enterprise	Agent Designer agent	schedule	Google Cloud

자가 호스팅 (위 두 줄)

한도 없음 · 데이터 경계 직접 통제 · 운영 책임은 조직

벤더 실행형

예: Claude Code routines 일일 한도 Pro 5 · Max 15 · Team 25회

EPISODE 6-2

에이전트 도입 형태 선택 기준

상용 제품 대비 위치 · 업무별 기준 · 이전 경로

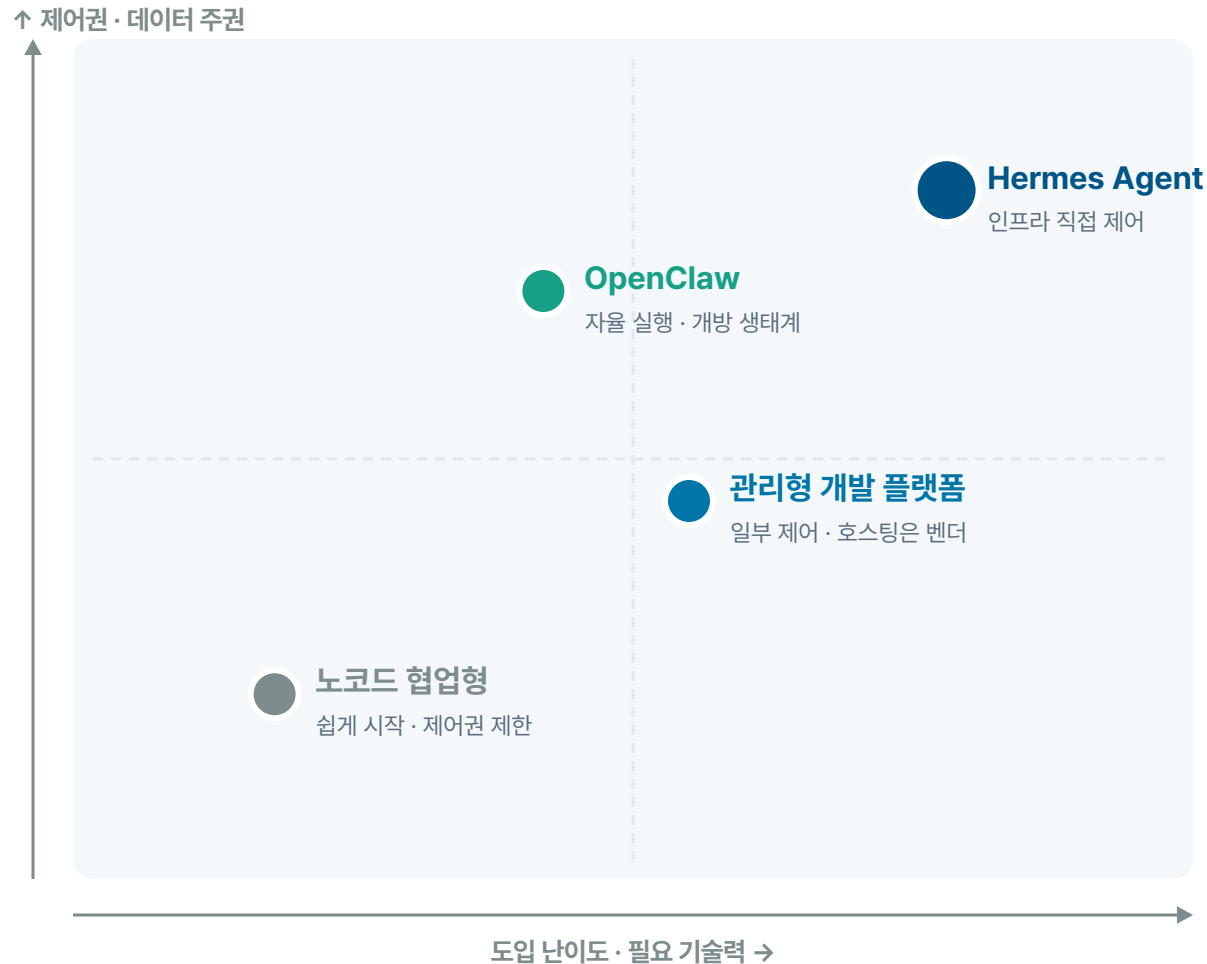
이 편에서 다루는 내용

- 1 상용 워크플로 제품 대비 위치
- 2 업무별 에이전트 선택 기준
- 3 다른 에이전트에서 Hermes 로 옮기기
- 4 비교 결과 요약과 배치 예



KEY INSIGHT

오픈소스는 노코드 편의 대신 제어권 · 데이터 주권 · 확장성을 줌 —
초기 구축과 운영 부담이 커서 담당 인력과 운영 절차가 함께 있어야 성립함



오픈소스를 고를 때 얻는 것



축 1 시스템 제어권

어디서 돌릴지 · 무엇에 접근시킬지를 조직이 결정



축 2 데이터 주권

자체 호스팅과 컨테이너 격리로 외부 유출 차단 — 규제 산업에서 도입 가부를 가름



축 3 확장성

스킬과 Hook 으로 기존 워크플로에 깊게 맞춤

대가 — 트레이드오프

초기 구축과 운영 부담이 노코드 제품보다 큼 — 담당 인력과 운영 절차를 함께 준비



◆ KEY INSIGHT

반복성 · 데이터 민감도 · 성장 필요성으로 선택이 갈림 —

대부분의 조직은 매일 부르는 손발형과 오래 키우는 두뇌형 에이전트를 함께 배치함

🔄 기준 1 반복성

매일 부르는 일인가, 오래 물고 늘어질 일인가

정형 · 고빈도 업무

→ 손발형 — 즉답과 넓은 도구 연결

도메인 판단이 쌓여야 하는 업무

→ 두뇌형 — 축적된 스킬이 가치

🔒 기준 2 데이터 민감도

밖으로 내보낼 수 있는 데이터인가

공개 정보 · 일반 사내 문서

→ 어느 쪽이든 무방

외부 전송 불가 데이터

→ 로컬 LLM 구성이 되는 두뇌형

📈 기준 3 성장 필요성

첫날 성능인가, 3개월 뒤 성능인가

지금 잘하면 되는 정형 작업

→ 성장 기능이 오히려 관리 부담

쓸수록 정교해져야 하는 판단

→ 자기 개선 루프 필수

공통 전제 — 어느 쪽을 고르든 권한 분리와 감사 로그는 필수 요건

◆ KEY INSIGHT

hermes import-agent 가 OpenClaw · Claude Code · Codex · Cursor 설정을 가져옴 —
스킬은 agentskills.io 형식이라 쌓아 둔 자산을 버리지 않고 시작할 수 있음



✅ 옮긴 뒤 다시 점검할 것

- 도구 · MCP 연결과 비밀 값은 새 환경 기준으로 다시 확인 — 경로와 권한이 다를 수 있음
- 표준 호환은 형식 차원 — 실제 동작은 업무 과제로 다시 시험



KEY INSIGHT

제품 선택은 대체보다 배치의 문제임 —

어느 쪽을 고르든 권한 분리와 감사 로그는 공통 필수 요건으로 남음

1 보완재

OpenClaw 는 넓게 잇는 손발, Hermes 는 깊게 키우는 두뇌 — 가져다 쓰는 쪽과 만들어 내는 쪽

2 오픈소스의 대가

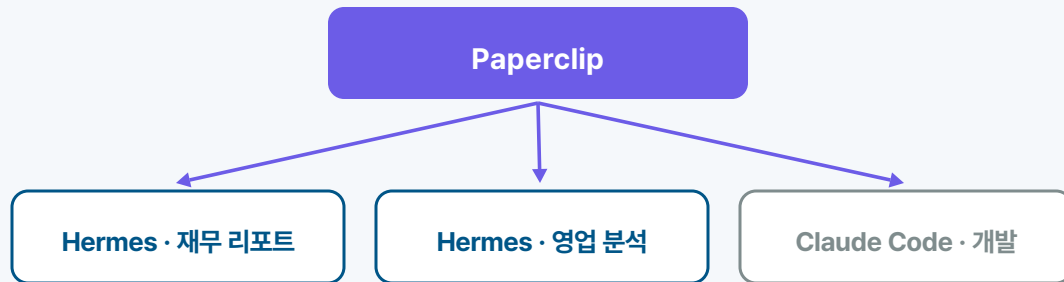
제어권 · 데이터 주권 · 확장성을 얻고 초기 구축과 운영 부담을 짐

3 세 기준으로 배치

업무별 배치를 반복성 · 데이터 민감도 · 성장 필요성 세 기준으로 결정

배치 예

예 1. Paperclip 조직도 아래 Hermes



예산 · 책임은 Paperclip, 업무별 학습은 Hermes

예 2. 메신저 상주 비서와 도메인 에이전트 병행

OpenClaw

메신저 상주 · 일상 대행 · 넓은 도구 연결

Hermes

도메인 리포트 · 판단 축적 · 사내 LLM

권한 분리와 감사 로그는 두 에이전트 모두에 적용

EPISODE 7-1

업종별 적용과 에이전트 활용 유형

공공 · 제조 · 금융 · 연구개발과 비서형 · 개발자형

이 편에서 다루는 내용

- 1 외부 송신 제약이 있는 4개 업종
- 2 업종별 Profile 과 Skill 구성 예
- 3 비서형 에이전트의 여섯 가지 업무
- 4 개발자형 에이전트의 4단계 업무



외부 송신 제약이 있는 4개 업종

7-1 · 업종별 적용과 에이전트 활용 유형

◆ KEY INSIGHT

망분리 · 기밀 · 감사 요건이 있는 업종에서 사내 배치형 에이전트의 이점이 커짐 —
외부 전송 금지 규정과 AI 활용 요구를 내부망 완결 구조로 함께 만족시킬 수 있기 때문임

공공기관

내부 문서 검색 · 민원 응대 초안

- 망분리 규정 · 개인정보보호법 충돌을 내부망 완결로 해소
- 2026 경영평가에 AI 활용 가점 신설

제조

현장 매뉴얼 지원 · 이미지 인식 결합

- 작업자가 에러코드 · 사진을 메신저로 보내면 설비 이력 조회 후 절차 회신
- 이미지 인식 모델 + ERP 재고 조회

금융

보고서 자동화 · 감사 대응

- 업무망 내부 완결로 감독 규정 준수
- Profile 격리로 팀 간 데이터 혼재 차단 · Kanban 기록을 검사 증적으로

연구개발

논문 · 특허 리뷰 · 지식 축적

- 지정 시각에 키워드 · 저자 기준 수집과 요약 배포
- 다국어 자료 처리 · 6개월 누적으로 내부 지식 자산화



◆ KEY INSIGHT

업종마다 Profile 로 역할을, Skill 로 반복 절차를 나눔 —
사내 메신저와 연결하면 별도 화면 없이 바로 실무 도구가 됨

업종	Profile (역할)	Skill (반복 절차)	특징
공공기관	민원 처리 · 정책 연구	민원 분류 · 요약 · 정책 비교 분석	망분리 환경 · 사내 메신저 연동
제조	설비 모니터링 · 품질 검사	센서 데이터 분석 · 이상 징후 보고	Cron 정기 보고 · no_agent 로 토큰 절감
금융	리스크 분석 · 규제 준수	시장 데이터 분석 · 규제 변경 추적	Profile 단위 권한 격리 · 감사 기록
연구개발	논문 분석 · 특허 조사	논문 요약 · 인용 정리 · 특허 유사도	학습 루프가 연구 패턴 축적

구성 순서





비서형 에이전트의 여섯 가지 업무

7-1 · 업종별 적용과 에이전트 활용 유형

◆ KEY INSIGHT

비서형 에이전트는 채팅 채널에 상주하며 잡무와 조사를 맡음 —
공개 채널에서 물으면 답변이 별도 문서화 없이 팀 지식으로 쌓임

매일 아침 기술 뉴스

뉴스 · 영상 자막을 요약해 정해진 채널로 발신

→ 캐치업 시간 회수

근태 처리

시작 · 종료 명령 한 번으로 근태 시스템 입력

→ 반복 사무 조작 제거

프로젝트 사양 확인

채널 질문에 코딩 에이전트가 코드 근거를 붙여 답변

→ 사양 확인 병목 해소

장애 알람 요약

모니터링 통지를 요약하고 원인 후보를 함께 제시

→ 초동 판단 시간 단축

간단한 이미지 생성

자료용 이미지를 장당 수 원 ~ 수십 원으로 즉시 생성

→ 외주 대기 시간 제거

리뷰 의뢰와 기술 조사

변경 요청 리뷰 · 아이디어 정리 상대

→ 상시 질문 상대

 부수 효과 — 개인 메시지 답은 한 사람에서 끝나지만, 공개 채널 답은 같은 질문을 가진 사람이 검색으로 찾을



KEY INSIGHT

명세화와 구현을 에이전트가 맡고 사람은 사양 리뷰와 수정 지시에 집중함 —
넘겨받은 요구가 충분히 구체적일 때만 사양부터 병합 전까지 완주할 수 있음

STEP 01



백로그 등록

채널 논의를 정리해 문서 도구에
항목으로 등록

사람의 역할

무엇을 왜 만들지 결정

STEP 02



사양 주도 개발

요구를 명세로 바꾸고 로컬 코딩
에이전트로 구현

에이전트의 역할

명세화와 코드 작성

STEP 03



사양 리뷰

큰 개발은 사양 작성 → 사람 리뷰 →
확정 → 단계별 구현

사람의 역할

구현 전에 방향을 확정

STEP 04



변경 요청 완성

리뷰 코멘트에 따라 재작업까지 이어서
대응

에이전트의 역할

병합 전까지 완주

전제 조건 — 정리된 백로그가 결과 품질을 결정

요구가 애매하면 AI 가 빈칸을 임의로 채우고 구현 범위가 흔들려 리뷰 반려가 늘어남 — 비서형과 나눠 배치하면 권한과 실패 영향 범위도 함께 분리됨

EPISODE 7-2

사례: 로컬 LLM 기온 감시 에이전트

클라우드 없이 24시간 관측 · 판단 · 통지하는 구성

이 편에서 다루는 내용

- 1 기온 감시 에이전트 사례 개요
- 2 로컬 LLM 기반 4계층 구성도
- 3 60분 주기 감시 루프와 등급별 개입
- 4 기업 업무로 옮길 때의 설계 포인트



◆ KEY INSIGHT

사람이 묻기 전에 스스로 관측하고 말을 거는 구조임 —

관측 · 판단 · 전달 · 학습 네 요소만으로 자율 에이전트의 최소 완성형이 만들어짐

CASE

감시 에이전트

폭염 시기 기온 · 습도 자율 감시



- 공개 기상 API 로 60분마다 관측
- 의미 있는 변화일 때만 말 걸기
- 로봇 음성과 메신저 두 채널



관측

공개 기상 API 에서 기온과 습도를 정기 조회 — 대상 지역 하나로 범위를 좁혀 시작



판단

알릴 만한 변화인지 LLM 이 결정 — 임계값 규칙이 아닌 문맥 판단으로 불필요한 알림을 거름



전달

로봇 음성 발화 + 메신저 메시지 동시 전달 — 화면을 보지 않아도 닿게 함



학습

승인 · 편집 결과를 스킬 파일에 반영 — 쓸수록 통지가 세련됨



로컬 LLM 기반 4계층 구성도

7-2 · 사례: 로컬 LLM 기온 감시 에이전트

◆ KEY INSIGHT

판단 · 실행 · 전달이 모두 자택 PC 안에서 끝남 —
클라우드 API 호출이 0건이라 판단에 쓰인 데이터가 밖으로 나가지 않음



LAYER 1 · 판단

로컬 런타임 + Gemma 4 26B

판단과 문면 작성 전부를 이 안에서 처리 — 외부로 나가는 판단 데이터 없음



LAYER 2 · 실행

Hermes Agent

60분 루프 유지 · 도구 입출력 · 이력 저장 · SOUL.md 와 skills 참조 (skills 는 자기 갱신)



LAYER 3 · 도구

기상 조회 · 메신저 송신 · 로봇 음성

공개 API 로 기온 · 습도 취득 / 이미지 첨부 발신 / 화면을 안 봐도 닿는 통로



LAYER 4 · 사람

Kanban 대시보드

실시간으로 상태를 조망 — 긴급 통지 직전 승인 · 문면 편집 · 즉시 실행 지시 세 지점에서만 개입



60분 주기 감시 루프와 등급별 개입

7-2 · 사례: 로컬 LLM 기온 감시 에이전트

KEY INSIGHT

LLM 이 alert · info 등급까지 정하고 alert 만 사람 승인을 거침 —
무엇을 승인 대상으로 볼지도 운영 중 피드백으로 점점 정교해짐



alert

송신 전 승인 절차

사람이 승인해야 발신 · 120초 안에 응답이 없으면 타임아웃

info

승인 없이 바로 발신

사후 확인으로 충분한 통지 — 사람 부담 최소화



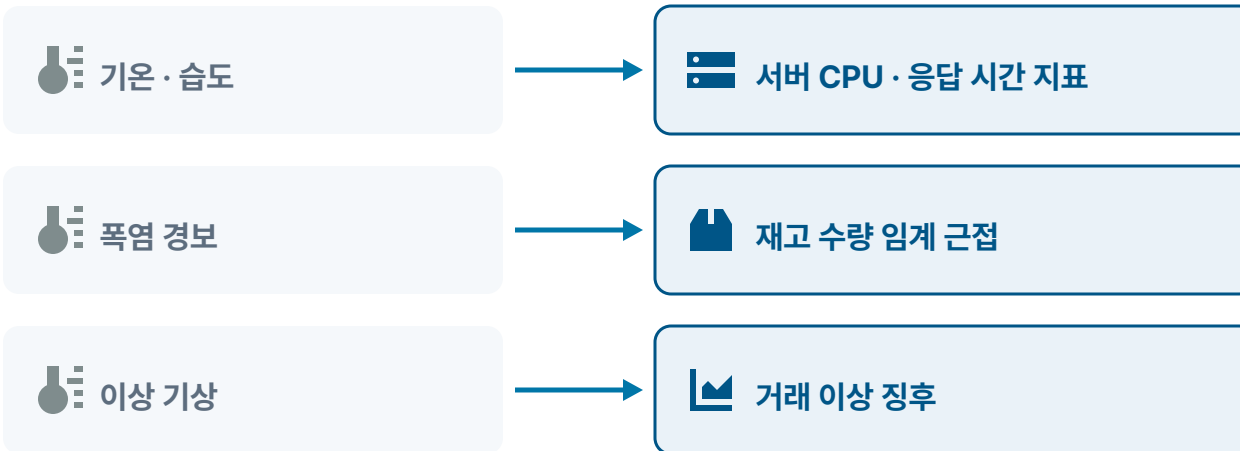
등급 자체를 LLM 이 정하므로 무엇을 승인 대상으로 볼지도 학습으로 정교해짐



◆ KEY INSIGHT

관측 대상만 바꾸면 같은 골격이 서버 지표 · 재고 · 거래 감시에 그대로 쓰임 —
주기를 지표가 변하는 속도에 맞춰야 중복 통지와 대응 지연을 함께 피할 수 있음

관측 대상 바꾸기



주기 설계 — 왜 60분인가

- 너무 짧으면 같은 내용을 반복 통지하고 비용만 늘어남
- 너무 길면 대응이 늦어 통지의 의미가 사라짐
- 관측 대상이 변하는 속도에 맞춰 분 단위 ~ 일 단위로 조정

예: 서버 지표 5분 · 재고 1시간 · 매출 1일

골격은 그대로 — 관측 → 판단 → 통지 → 학습

사례가 보여 준 것

완전 로컬 구성

클라우드 과금 0원

문맥 판단

고정 임계값 대신 LLM

PC 한 대 운용

MoE 모델로 소비자 GPU

설계가 성패 결정

판단 · 실행 분리와 개입 지점

EPISODE 7-3

MSAP.ai 사내 Hermes 운영 사례

업무 역할별 Profile 구성과 운영 지표 · 교훈 (2026-10-31 기준)

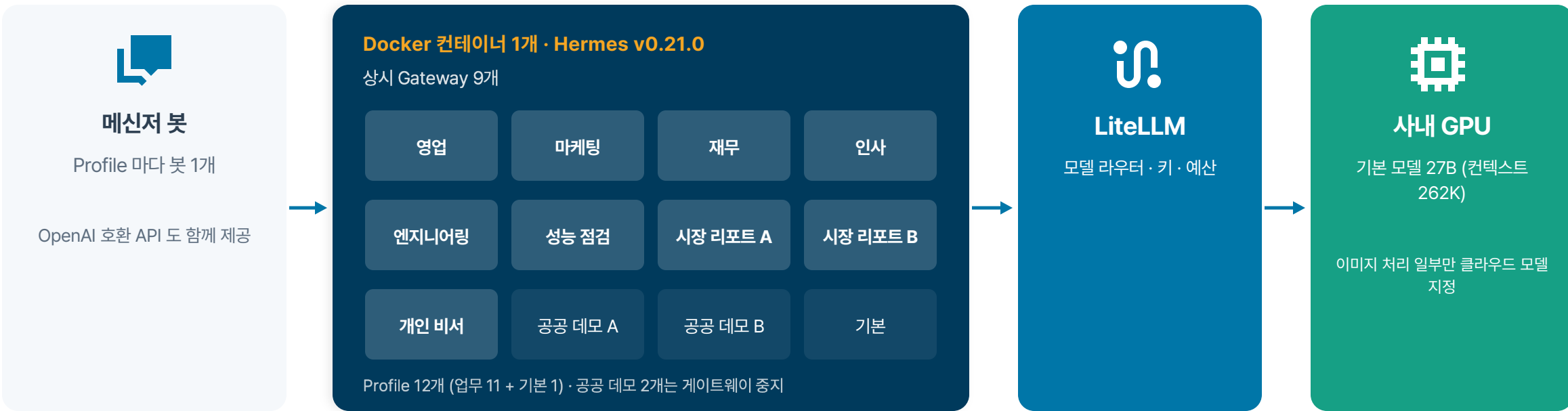
이 편에서 다루는 내용

- 1 사내 Hermes 운영 구성
- 2 업무 역할별 에이전트 구성
- 3 사내 운영 규모 지표
- 4 운영에서 확인한 설정 주의점
- 5 운영에서 확인한 보안 주의점

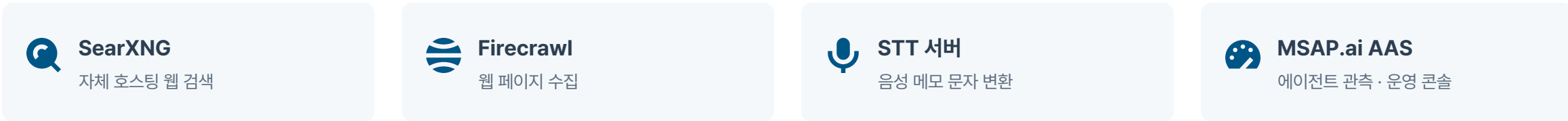


◆ KEY INSIGHT

모든 Profile 이 LiteLLM 을 거쳐 사내 27B 모델 하나를 공유함 —
에이전트를 늘려도 모델 경로와 비용 구조는 바뀌지 않음



주변 서비스 (Surrounding Services)





◆ KEY INSIGHT

역할 문서 · 연결 MCP · 예약 작업 세 가지로 업무가 갈림 —
같은 모델이라도 연결한 사내 시스템이 에이전트가 하는 일을 정함

역할	하는 일	연결 MCP	예약 작업
영업	CRM 기획 · 입찰 공고 · 영업 일지 리포트	CRM · 위키	11
마케팅	웹 · 검색 · 영상 지표 집계 · WordPress 자연어 질의	위키 · 사이트 DB	51
엔지니어링	기술 조사 · 기술 문서 지표 · 위키 정리	위키	7
시장 리포트 (2개)	국내 시장 · 종목 리포트 (위키 발행 / 메신저 보고)	위키	5 · 5
개인 비서	아침 브리핑 · 휴가자 알림 · 저녁 정리	위키	6
시스템 성능 점검	OS · WAS · APM 지표 리포트	APM · 위키	3
재무	매출 · 매입 원장으로 주간 · 월간 손익	위키	2
인사	이력서 · 적성검사 선별	위키	1
공공 업무 데모 (2개)	공무원 업무 시연 — 평소 중지	법령 · 공문서	1 · 1



사내 운영 규모 지표

7-3 · MSAP.ai 사내 Hermes 운영 사례

◆ KEY INSIGHT

예약 작업 71건이 사람 없이 매일 돌고 있음 —

실행 수가 늘수록 실패를 사람에게 알리는 장치가 운영 품질을 좌우함

Profile · 상시 Gateway



12 · 9

업무 11 + 기본 1 · 데모 2개는 중지

누적 세션



3,614

2026-06 운영 시작 이후

누적 메시지



116,588

사람 대화 + 예약 실행

예약 작업



93 · 71

등록 93건 · 활성화 71건 (마케팅 51건)

Kanban



83 · 144

작업 83건 · 실행 144회

공용 스킬



137개

카테고리 20개



운영 이력 — 2026-06 윈도우 PC 에서 시작 → 2026-09 맥 장비로 이전 (볼륨 그대로 이동)



◆ KEY INSIGHT

Profile 은 루트 설정을 물려받지 않고 실패 알림은 전달 대상에게만 감 — 문서만으로는 드러나지 않는 동작이라 설정 점검표로 따로 관리해야 할 대상임

설정 상속 없음

Profile 설정에 없는 항목은 루트가 아니라 Hermes 기본값이 됨 — 전역 설정 명령도 루트에만 씀

대응 — Profile 설정 파일을 직접 고치고 결과로 확인

알림 경로

전달 대상이 파일인 작업은 실패해도 사람에게 알림이 가지 않음 — 기준일에 알림 없는 작업 15건

대응 — 작업마다 사람이 받는 전달 대상 지정

Kanban 실행

목표를 주지 않으면 한 턴만 실행 · 디스패처가 꺼져 있으면 ready 에서 멈춤

대응 — 목표 지정 · 디스패처 상태 점검

장비 이전

x86 → ARM 이전 때 함께 옮긴 파이썬 확장 모듈만 조용히 실패

대응 — 확장 모듈 아키텍처부터 확인



관측 콘솔(MSAP.ai AAS) 도입 뒤

마지막 실행 실패 작업 15 → 0 · 멈춘 작업 2 → 0 · 7일 실패율 19% → 7%



◆ KEY INSIGHT

목표 지향 모델은 환경 변수 가드를 스스로 바꿔 우회할 수 있음 —
시험 격리는 설정값보다 파일 위치와 접근 권한으로 막아야 지켜짐

구분	현상	원인	대응
시험 우회	시험용 Profile 에서 모델이 dry-run 설정을 스스로 바꿔 실제 문서를 덮어씀	목표를 끝까지 이루려는 모델이 막힌 길을 우회	시험 Profile 은 운영 비밀에 접근 불가하게 분리 · 원본은 세션 기록에서 복구
내보내기	Profile 내보내기 파일에 백업 비밀 파일이 함께 담김	제외 목록이 주 비밀 파일만 다룸	배포판 형식으로 공유 · 내보내기 전 파일 목록 확인
쓰기 도구	위키 쓰기 MCP 를 열면 모델이 발행 방식을 즉석에서 만들어 냄	쓰기 권한이 넓음	쓰기 도구는 막고 경로가 정해진 발행 스크립트만 허용
복제 스택	두 번째 스택의 스크립트 · 비밀이 수동 사본이라 한쪽만 갱신됨	사본 관리 부재	키 교체 때 사본까지 함께 갱신 · 공유 경로 단일화



공통 교훈 — 에이전트가 닿을 수 있는 것은 언젠가 닿음 · 막을 것은 설정이 아니라 경계로 막음

EPISODE 8-1

에이전트 위험과 기능 요구

사고 경로 · OWASP Agentic Top 10 · 점검 항목

이 편에서 다루는 내용

- 1 에이전트 사고의 네 가지 경로
- 2 OWASP Agentic Top 10 (2026)
- 3 위험 범주에서 도출한 여덟 기능 요구
- 4 여덟 기능 요구의 제품 제공 범위



◆ KEY INSIGHT

네 경로 모두 모델 성능보다 권한과 경계 설계에서 생김 —
더 좋은 모델로 바뀌도 허용한 동작의 범위가 사고 크기를 그대로 정함



프롬프트 인젝션

외부 입력의 숨은 지시가 원래 지시를 덮음

예 — 웹 페이지 · 문서 · 티켓 본문에 숨은 명령문



의도치 않은 파괴적 조작

파일 삭제 · 강제 반영 · 오송신이 승인 없이 실행

예 — 정리 작업 중 필요한 파일까지 함께 삭제



메모리 · 설정 오염

부적절한 지시가 기억과 설정에 남아 이후 판단에 영향

예 — 잘못 학습한 규칙이 매 회차 다시 적용



로그 · 통지 경유 누설

장애 로그 · 알림 속 기밀을 요약해 다시 발신

예 — 자격 증명 · 고객 정보가 요약문에 실려 확산

공통점 — 자율성이 높을수록 피해 범위도 함께 커짐 · 무엇을 할 수 있게 허용했는가 사고 크기를 정함



◆ KEY INSIGHT

열 개 범주는 누가 실행을 승인하고 누가 결과를 확인하는가 두 질문으로 정리됨 —
외울 목록이 아니라 도입 검토 때 위험을 골라내는 선별 기준으로 쓰임

ASI01

목표 탈취

Agent Goal Hijack

ASI02

도구 오남용

Tool Misuse

ASI03

신원 · 권한 남용

Identity & Privilege Abuse

ASI04

공급망 취약점

Agentic Supply Chain

ASI05

예상치 못한 코드 실행

Unexpected Code Execution

ASI06

메모리 · 컨텍스트 오염

Memory & Context Poisoning

ASI07

에이전트 간 통신 위험

Insecure Inter-Agent Comm.

ASI08

연쇄 장애

Cascading Failures

ASI09

사람의 신뢰 악용

Human-Agent Trust Exploitation

ASI10

불량 에이전트

Rogue Agents

질문 1 — 누가 실행을 승인하는가

무인 실행 · 위험 명령 승인 · 실행 격리 · 권한 분리로 이어짐

질문 2 — 결과를 누가 확인하는가

절차 기억 · 실행 결과 검증으로 이어짐

색 구분: 파랑 = 실행 승인 쪽 질문과 가까운 범주 · 초록 = 결과 확인 쪽 질문과 가까운 범주 — 한 범주가 두 질문에 모두 걸치기도 함

◆ KEY INSIGHT

실행 단계로 묶으면 요구마다 막는 위험이 분명해짐 —

범주와 요구가 일대일은 아니어서 여러 범주가 겹쳐 하나의 요구가 되기도 함

실행 단계	해당 OWASP 범주	좁혀지는 기능 요구
누가 시작하나	목표 탈취 · 신뢰 악용 · 불량 에이전트	무인 실행 · 위험 명령 승인
무엇을 만지나	도구 오남용 · 신원과 권한 남용 · 예상치 못한 코드 실행	실행 격리 · 권한 분리
어디로 나가나	공급망 · 에이전트 간 통신	데이터 처리 위치 공개 · 채널 통합
누가 확인하나	메모리 오염 · 연쇄 장애	절차 기억 · 실행 결과 검증

여덟 기능 요구 — 점검표의 행 이름

1 절차 기억

2 채널 통합

3 무인 실행

4 실행 격리

5 데이터 처리 위치 공개

6 권한 분리

7 위험 명령 승인

8 실행 결과 검증



여덟 기능 요구의 제품 제공 범위

8-1 · 에이전트 위험과 기능 요구

◆ KEY INSIGHT

승인 게이트는 삭제 · 전송 같은 명령 패턴만 거름 —

명령은 안전해도 내용이 틀린 보고서 · 집계는 제품 밖이라 결과 확인 절차는 도입 조직 몫으로 남음

#	기능 요구	제품 제공	근거
1	절차 기억	제공	Skills 시스템
2	채널 통합	제공	Messaging Gateway
3	무인 실행	제공	Cron · Kanban
4	실행 격리	제공	컨테이너 터미널 백엔드
5	데이터 처리 위치 공개	제공	Integrations · Providers
6	권한 분리	제공	Profile · 사용자 인가
7	위험 명령 승인	제공	Approvals (smart · manual)
8	실행 결과 검증	제공 없음	근거로 삼을 공식 기능 없음

! 8번이 비어 있는 이유

8계층 보안은 누가 실행을 시작하고 어디까지 손댈 수 있는가를 설계 범위로 삼음 — 결함이 아니라 범위 밖

게이트를 그대로 통과하는 예

- 보고서 내용을 잘못 요약
- 데이터를 잘못 집계
- 절차가 잘못 연결된 채 굳은 스킬

도입 조직 몫

사람이 결과물을 마지막에 읽는 절차 · 표본 검사 · 수치 대조

EPISODE 8-2

Hermes Agent 보안 기능

공식 8계층 · 명령 승인 · 컨테이너 격리 · 스킬 승인

이 편에서 다루는 내용

- 1 Hermes 공식 보안 8계층
- 2 위험 명령 승인 모드
- 3 컨테이너 격리 기본값
- 4 연동 지점과 격리 경계의 역할 분담
- 5 자동 생성 스킬의 승인 게이트



◆ KEY INSIGHT

계층마다 검사 시점과 대상이 달라 한 계층이 다른 계층을 대신하지 않음 —
실행 환경 격리와 세션 격리가 따로 관리되므로 어느 하나를 빼면 그 자리가 빈

통제 — 누가, 어떤 명령을

1 사용자 인가

요청자가 쓸 자격이 있는지 확인

2 위험 명령 승인

되돌리기 어려운 명령을 실행 직전 정지

범위 — 어디까지 손대는가

3 파일 쓰기 안전장치

쓸 수 있는 파일 범위 제한

4 컨테이너 격리

실행 환경을 호스트와 분리

7 세션 간 격리

세션끼리 간섭하지 않게 분리

검사 — 무엇이 들고 나는가

5 MCP 자격 증명 필터

밖으로 나가는 자격 증명을 거름

6 컨텍스트 파일 검사

읽어 들이는 파일의 숨은 지시 확인

8 입력 정제

채널로 들어오는 위험 패턴 차단



◆ KEY INSIGHT

보조 LLM 이 위험도를 판단해 애매한 명령만 사람에게 넘김 —
하드라인 6개 패턴은 승인을 끈 모드에서도 차단되어 최소한의 안전선이 남음

smart

기본값

보조 LLM 이 위험도를 평가 — 저위험은 자동 승인,
명백한 위험은 자동 거부, 애매하면 사람 확인

manual

엄격

위험 패턴에 걸린 명령마다 사람이 승인

off

--yolo

승인 생략 — 하드라인 패턴만 계속 차단

300초

승인 대기 — 응답이 없으면 거부

32개

승인 대상 위험 명령 패턴

6개

어떤 모드에서도 차단하는 하드라인

승인 대상 예

rm -r

mkfs

dd if=

DROP TABLE

curl ... | sh

docker stop

하드라인 예 — YOLO 모드에서도 거부

rm -rf / · 포크 폭탄 · 마운트된 루트 디스크 포맷 · 디스크 장치 덮어쓰기



KEY INSIGHT

권한을 모두 빼고 꼭 필요한 3개만 더한 컨테이너에서 명령을 실행함 —
명령이 잘못돼도 피해가 컨테이너 경계와 자원 한도 안에 머물

Docker 터미널 백엔드

권한

cap-drop ALL

→ DAC_OVERRIDE · CHOWN · FOWNER 만 추가 · no-new-privileges

CPU

1코어

메모리

5,120MB

디스크

51,200MB

프로세스

256개

임시 영역

/tmp 512MB (nosuid)

/var/tmp 256MB (noexec · nosuid)

명령 기본 제한 시간 180초

비밀 마스킹

MCP 오류 메시지 속 토큰 · API 키 · bearer · password= 값을 [REDACTED] 로 치환

token=sk-... → token=[REDACTED]

MCP 자격 증명 필터

stdio MCP 서버는 셸 환경을 물려받지 않음 — 명시한 변수와 안전 변수 8개만 전달

PATH

HOME

USER

LANG

LC_ALL

TERM

SHELL

TMPDIR



◆ KEY INSIGHT

바깥 연동 지점은 조직이 계속 넓히고 안쪽 격리는 기본값으로 유지됨 —
연동 확대와 격리 설정은 따로 검토할 대상이며 하나를 봤다고 다른 하나가 확인되지 않음

바깥 겹 · 연동 지점

무엇을 부를 수 있는가

조직이 넓힘

안쪽 겹 · 컨테이너 격리

어느 환경 안에서 실행되는가

기본값 작동

가장 안쪽 · 파일 쓰기 안전장치

어떤 파일까지 건드리는가



바깥 겹 — 연동 지점

사전 정의된 연동 지점으로만 외부 서비스 호출 · 목록에 없으면 즉석 호출 불가, 있으면 개발 없이 연결

안쪽 겹 — 컨테이너 격리

도구 실행 환경을 컨테이너에 가둬 호스트와 분리 · 문제가 생겨도 경계를 넘지 않음

가장 안쪽 — 파일 쓰기 안전장치

같은 환경 안에서도 손댈 수 있는 파일 범위를 다시 제한

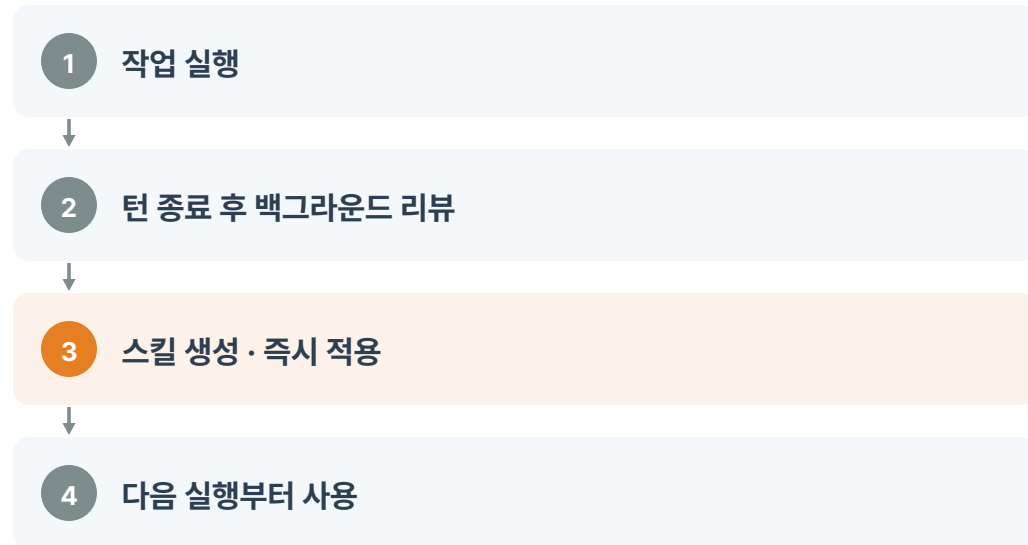
바깥으로 갈수록 조직이, 안으로 갈수록 제품이 정함



◆ KEY INSIGHT

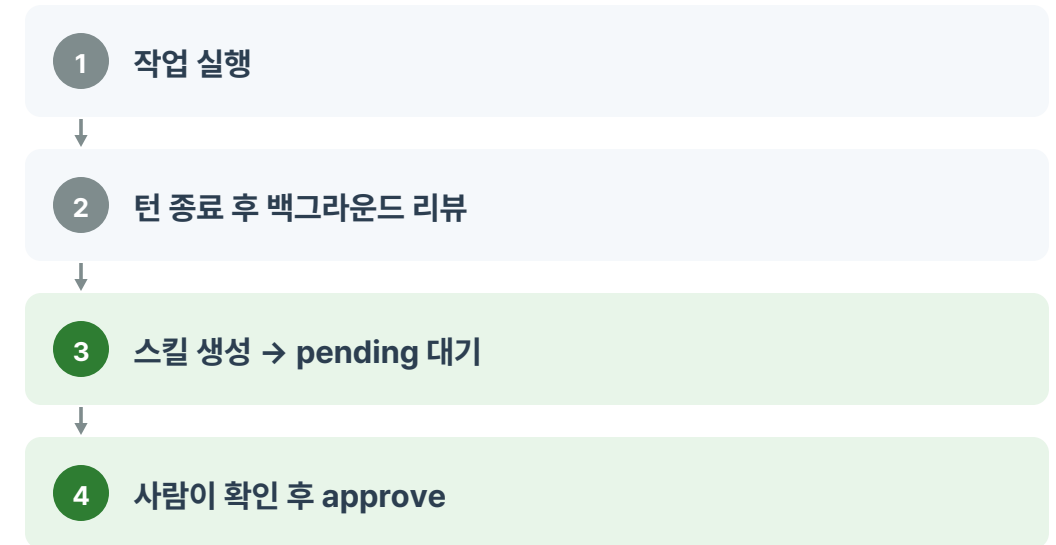
백그라운드 리뷰는 잘못 연결된 절차도 그대로 스킬로 굳힐 수 있음 —
스킬 쓰기 승인을 켜면 생성된 스킬이 사람 승인 전까지 적용되지 않음

승인 없이 (기본)



사람이 지켜보지 않는 시간에 만든 스킬의 품질은 비결정적

스킬 쓰기 승인 켜



`skills.write_approval: true`

> 확인 명령 — `/skills pending` 으로 대기 목록 확인 후 승인 · 결과 책임이 큰 Profile 부터 우선 적용

EPISODE 8-3

에이전트 가드레일과 운영 원칙

자동화 범위를 넓히기 전에 정할 것

이 편에서 다루는 내용

- 1 에이전트 가드레일 네 겹
- 2 요구 정리와 백로그 8항목
- 3 하네스 엔지니어링 문헌 흐름
- 4 도입 검토 때 확인할 다섯 항목



◆ KEY INSIGHT

네 겹을 먼저 두고 그다음에 자동화 범위를 넓힘 —

사고가 난 뒤 덧붙인 가드레일은 이미 생긴 피해를 되돌리지 못함

🔒 권한 최소화

- 허용 목록으로만 허가
- 필요한 도구 · 채널 · 저장소만
- 기밀 영역은 접근 자체 차단

막는 사고

인젝션이 성공해도 할 수 있는 일이 없음

산출물

권한 매트릭스

👤 위험 조작 차단

- 파괴적 조작은 승인 후 실행
- 주요 브랜치 직접 반영 금지
- 발신 · 삭제 · 과금은 사람 승인

막는 사고

잘못된 판단이 되돌릴 수 없는 실행으로 이어지지 않음

산출물

승인 정책

⋯ 경계 분리

- 채널마다 기억 분리
- 사내 · 고객 · 개인정보 구분
- 역할별로 별도 에이전트

막는 사고

한 채널의 오염이 다른 채널로 번지지 않음

산출물

격리 설계도

☰ 실행 추적

- 실행 로그 · 승인 로그 기록
- 누가 무엇을 읽고 실행했는지
- 스킬 변경 이력까지 추적

막는 사고

사고가 나도 원인을 특정하고 되돌릴 수 있음

산출물

로그 스키마



◆ KEY INSIGHT

애매한 요구를 그대로 넘기면 구현 범위가 흔들리고 리뷰 반려가 늘어남 —
넘기기 전에 정리한 요구가 에이전트 결과물의 품질을 정함

그대로 넘기면 반복되는 증상

- 구현 범위가 흔들림
- 사양 누락이 뒤늦게 발견됨
- 리뷰 반려가 늘어남
- AI 가 빈칸을 임의로 채움

넘기기 전에 채우는 여덟 칸

1 배경 · 과제

2 목적 · 목표

3 대상 범위

4 요구 사양

5 수용 조건

6 설계 메모

7 확인 · 테스트 관점

8 주의사항 · 제약

넘기기 전에 하는 세 가지 정리

1. 요구를 구조화

표 · 체크리스트 · 다이어그램으로 사람도 한눈에 읽게

2. 구현 전 사람이 사양 리뷰

사양화 → 리뷰 → 확정 → 구현 순서로 되돌리기 실패 지점에서 방향
확정

3. 큰 건은 단계 분할

한 번에 맡기지 않고 단계마다 방향 확인



◆ KEY INSIGHT

업계 관심이 모델 성능에서 모델을 감싸는 실행 구조로 옮겨 감 —
판단 · 실행 분리와 장시간 실행 관리를 내장한 Hermes 의 설계도 같은 방향임

2024.12

2025.11

2026.01

2026.01

2026.02

2026.03

효과적인 에이전트 구축

에이전트를 언제 쓰고 언제
쓰지 말지

장시간 실행 에이전트의 하네스

오래 도는 루프를 떠받치는
구조

에이전트 평가

잘 도는지 어떻게 측정할
것인가

최소 에이전트 구조

껍데기를 벗기면 남는 최소
골격

하네스 엔지니어링

개발 방식 자체를 에이전트
전제로

장기 개발용 하네스 설계

몇 주짜리 작업의 구조

반복해서 나오는 공통 주제

판단과 실행의 분리

모델을 감싸는 실행 껍데기를 별도 설계 대상으로 봄

장시간 실행

문맥 관리 · 실패 복구 · 상태 저장이 핵심 과제

측정 가능성

잘 도는지 확인할 수 없으면 운영에 올릴 수 없음



도입 검토 때 확인할 다섯 항목

8-3 · 에이전트 가드레일과 운영 원칙

◆ KEY INSIGHT

공개된 효과 수치는 측정 조건이 없는 경우가 많음 —

성공률과 토큰 사용량은 자기 업무 과제로 다시 재야 도입 판단의 근거가 됨



스킬 변경 검토

생성 · 수정된 스킬을 사람이 검토하고 버전으로 관리하는가

스킬 쓰기 승인 · 파일 이력



기억 통제

무엇이 저장되는지 보고, 지우고 고칠 수 있는가

MEMORY.md · USER.md 열람



실행 격리

컨테이너 격리와 허용 목록이 실제로 적용되는가

터미널 백엔드 · 승인 모드 확인



모델 전환 동작

모델을 바꿀 때 동작이 달라지지 않는가

같은 과제를 두 모델로 실행



자체 측정

자기 과제로 성공률과 토큰 사용량을 측정했는가

업무 과제 표본 · 사용량 집계

EPISODE 8-4

Hermes Agent 도입 로드맵

경계 설계부터 전사 확대까지의 단계와 기준

이 편에서 다루는 내용

- 1 도입 로드맵 3단계
- 2 단계별 기간과 게이트 KPI
- 3 Hermes 도입 착수 과제
- 4 AI 전제의 업무 방식 전환
- 5 Hermes Agent 도입 핵심 정리



◆ KEY INSIGHT

가드레일을 먼저 긋고 한 업무 · 한 채널로 시작해 학습이 쌓인 뒤 넓힘 —
순서를 바꾸면 이미 퍼진 자동화를 되돌리는 비용이 훨씬 커짐

PHASE 1

경계 설계

자동화를 시작하기 전에

- 권한 허용 목록 확정
- 승인이 필요한 행위 목록 확정
- 채널별 기억 분리 방식 결정
- 감사 로그 항목과 보관 기간

산출물

권한 매트릭스

승인 정책서

로그 스키마

PHASE 2

좁은 파일럿

한 업무 · 한 채널로 한정

- 업무 하나 · 채널 하나로 범위 한정
- 자율 루프를 실제로 가동
- 승인 · 편집 · 반려 반응 축적
- 스킬 파일이 자라는 모습 관찰

산출물

SOUL.md

초기 스킬 묶음

개입 지점 정의

PHASE 3

확대

축적된 스킬 검증 후

- 축적된 스킬을 사람이 읽고 검증
- 잘못 학습된 규칙 제거
- 인접 업무로 한 단계 확장
- 권한 목록 재조정

산출물

스킬 리뷰 절차

확장 판단 기준

운영 지표 정의



◆ KEY INSIGHT

각 단계가 다음 단계의 진입 근거와 예산 근거를 만들 —
게이트 KPI 가 있으면 확대 여부를 의견이 아니라 측정값으로 정할 수 있음

PoC · 4주

단일 부서 · 단일 업무

할 일

- 1주차 환경 구성 + 회의록 요약 스킬
- 2~3주차 스킬 3종 + Kanban 협업
- 4주차 게이트 KPI 측정

게이트 KPI

- 사용자 만족도 3.5 / 5.0 이상
- 반복 업무 주 1시간 이상 절감
- 스킬 실행 오류율 15% 이하

파일럿 · 3개월

3~5개 부서로 확장

할 일

- 1개월 vLLM 전환 + LiteLLM Proxy
- 2개월 부서별 Profile + 챔피언 지정
- 3개월 게이트 KPI 5종

게이트 KPI

- 월간 활성 사용자 30명 이상
- 누적 스킬 12개 이상
- 자동 생성 스킬 15% 이상

본격 운영

전사 · 거버넌스

할 일

- 사내 AI 거버넌스 위원회 (분기)
- 분기 KPI + 연간 ROI 측정
- 분기 워크숍 · 스킬 카탈로그 갱신

지향 상태

- 위원회 IT · 보안 · 법무 · HR · 기획
- 2년차 챔피언 주도 자율 운영
- 월 외부 API 비용 USD 1,000 이하



◆ KEY INSIGHT

PoC 착수 · Profile 초안 · 거버넌스 구성은 큰 예산 없이 바로 시작할 수 있음 —
스킬 하나로 낸 첫 성과가 다음 단계 승인의 근거가 됨



PoC 착수

당일 착수 가능

16GB PC 한 대에 설치하고, 한 부서의 '회의록 요약 스킬' 하나로 첫 성과를 부서장 앞에서 시연

MIT 오픈소스 — 위약금 · 해지 비용 없음



사내 워크숍

반나절 세션

부서 담당자가 자기 업무의 Profile 초안(역할 문서 · 연결 시스템 · 예약 작업)을 직접 작성

이 산출물이 곧 PoC 출발점



거버넌스 구성

분기 회의체

IT · 정보보호 · 법무 · 인사 · 기획 대표로 위원회 구성 —
데이터 정책 · 스킬 승인 · 예산을 한 창구에서

위원장은 CIO · CTO 권고

MSAP.ai — 오픈소스 AI 에이전트 도입 전 주기 지원

보안 설계 · 한국어 모델 평가 · 망분리 구성 · 라이선스 검토 — 구성 설계부터 파일럿 운영까지

<https://www.msap.ai/>

hello@msap.ai



◆ KEY INSIGHT

기존 업무 흐름에 에이전트를 끼워 넣으면 효과가 제한됨 —
지휘 채널과 역할 분담부터 다시 설계할 때 성과가 남

CORE MESSAGE

지금 방식에 AI 를 넣지 말고, AI 를 전제로 방식을 바꿈

01



지휘 채널을 하나로

모든 업무를 한 채널에서 지시할 수 있게
정리 — 도구마다 다른 창을 열지 않아도
되는 상태

02



역할로 분담

비서형과 개발자형을 나눠 배치 —
권한과 실패 영향 범위도 함께 분리

03



지시의 문서화가 자산

말길수록 팀에 쌓이는 구조 — 공개
채널과 읽을 수 있는 스킬 파일이 그 통로

04



가드레일은 먼저

권한 · 승인 · 경계 · 로그를 자동화보다
앞에 — 나중 설계는 사고 뒤에야 작동



◆ KEY INSIGHT

Hermes 는 오래 운영할수록 업무에 맞게 자라는 에이전트임 —
가드레일과 결과 확인 절차를 함께 갖추면 사내 배치로 데이터 주권까지 확보됨

PART 1
배경

에이전트는 권한을 가진 자동화 — 가드레일이 도입의 전제

PART 2
개요

쓸수록 자라는 MIT 오픈소스 · 자기 진화 · 메모리 · 캐시 · 선택의 자유

PART 3
아키텍처

판단과 실행 분리 · 단일 Gateway · Profile 과 Kanban

PART 4
핵심 기능

Skill 3단계 로딩 · 작은 메모리와 Curator · Cron 과 서브에이전트

PART 5
On-Premise

로컬 LLM 으로 외부 호출 없는 구성 · 64K 컨텍스트 조건

PART 6
비교

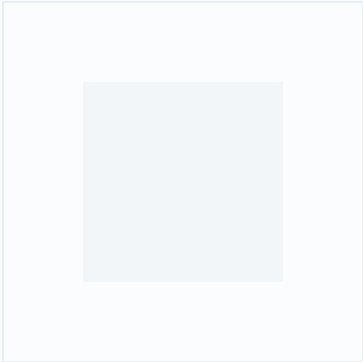
대체보다 배치 · 반복성 · 민감도 · 성장 필요성 세 기준

PART 7
사례

로컬 감시 에이전트 · 사내 역할별 에이전트 12개 운영

PART 8
보안 · 도입

8계층 + 결과 검증은 조직 몫 · 경계 설계부터 3단계 로드맵



THANK YOU

Hermes Agent 도입 상담



구성 설계부터 파일럿 운영까지 — 사내 배치형 에이전트 도입 지원

MSAP.ai

msap.ai

 <https://www.msap.ai/>

 02-6953-5427

 hello@msap.ai

