UNIT 2.7 · 헤르메스 스킬 이해하기

스킬 — 비서가 필요할 때 꺼내 읽는 업무 매뉴얼

Sophie에게 '필요할 때만 펼치는 매뉴얼'을 쥐여주기 · Section 2 · Hermes 기본 운영

스킬이란 무엇인가

ON-DEMAND 지식 문서

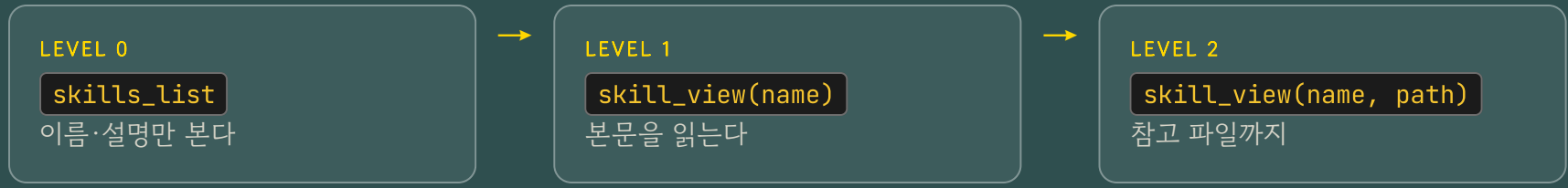
- 스킬은 **평소엔 목록·설명만** 있다가, 필요할 때 본문이 로드되는 **on-demand 지식 문서**입니다.
- 에이전트(Sophie)의 **"업무 매뉴얼"** — 사람이 따로 시키지 않아도 스스로 꺼내 읽습니다.
- 모든 스킬의 단일 진실원천: `~/.hermes/skills/`

책장 = skills 목록 (평소) → 한 권만 펼침 = 로드 (필요할 때)

자동 로드의 원리 — Progressive Disclosure

이 유닛의 핵심 한 장

스킬을 부르는 **기본 방식은 에이전트의 자동 로드**입니다. 토큰을 아끼려고 3단계로, **에이전트가 스스로** 필요를 판단해 펼칩니다.



"에이전트는 진짜 필요할 때만 전문(全文)을 읽습니다 — 사람이 슬래시로 부르지 않아도."

스킬은 어디에, 어떻게 생겼나

폴더 구조

```
~/.hermes/skills/<카테고리>/<스킬>/
├── SKILL.md          (필수 — 지침 본문)
├── references/       (참고 문서, 선택)
├── scripts/          (실행 스크립트, 선택)
├── templates/        (양식, 선택)
└── assets/           (이미지 등, 선택)
```

- 스킬 하나 = **폴더 하나**. 그 안의 `SKILL.md`가 심장입니다.

SKILL.md 한 장 뜯어보기

스킬은 코드가 아니라 잘 쓴 문서

```
# ── frontmatter (--- 로 감싸는 머리말) ──
name: daily-brief # 식별자 (슬래시 이름으로도 쓰임)
description: 아침 브리핑 ... # 언제 쓰는지 — 자동 매칭의 트리거
# ── 본문 4섹션 ──
#   When to Use   언제 쓰나
#   Procedure     절차
#   Pitfalls      함정
#   Verification  검증
```

- 위 **frontmatter**(name · description) + 아래 **본문 4섹션**. 잘 쓴 설명문이 곧 잘 도는 스킬.

스킬 vs 도구(tool) — 언제 무엇

스킬 · SKILL

지침 + 기존 CLI

지침을 글로 적고, 이미 있는 터미널·명령으로 표현되면 → **스킬**.

도구 · TOOL

코드로 실행

API 키 흐름·바이너리·스트리밍·정밀 실행이 필요하면 → **도구(tool)**.

대부분의 업무 자동화는 '스킬'로 충분합니다.

스킬은 어떻게 불려나오나

자동 로드가 기본 · 슬래시는 수동 호출

기본 · 자동 로드

에이전트가 "이 일엔 이 매뉴얼"이라고 **스스로 판단**해 꺼냅니다 (S3).

보조 · 슬래시 수동

`/스킬이름` 은 그 SKILL.md를 메시지로 실어 보내 '로드'하는 것 — **실행이 아닙니다**.

에이전트 자동 판단 ↘ 로드 (≠ 실행) ↙ 사람이 /plan 입력

플랫폼 주의: 슬랙만 네이티브 슬래시 50개 제한 → 거기선 자연어나 `/hermes` 로.

여러 스킬을 한 번에 — 묶음 & reload

HERMES BUNDLES · /RELOAD-SKILLS

- **묶음**(`hermes bundles`): 늘 같이 쓰는 스킬 셋을 **슬래시 하나** `/<묶음>` 로 동시 로드.
- 내장 '번들 스킬'과는 **다른 개념** — 이건 **내가 직접 묶는** 것 (`list/show/create/delete/reload`).
- `/reload-skills`: 새로 설치·이동한 스킬을 **재시작 없이 즉시 재스캔**. "설치했는데 안 보임"의 해결책.

내 비서는 뭘 갖고 있나

여기서부터 라이브 데모

```
$ hermes -p sophie skills list          (예시 화면)
NAME              CATEGORY      SOURCE    TRUST    STATUS
google-workspace  productivity  builtin   builtin  enabled
plan              planning      builtin   builtin  enabled
himalaya          email         builtin   builtin  enabled
...
— 90 builtin · 6 local · 0 hub — 96 enabled
```

- 컬럼 읽는 법: **Name** · **Category** · **Source** · **Trust** · **Status**.
- 슬라이드 대신 **실제 화면**으로 — 여기서부터 실습.

builtin · local · hub, 그리고 '통합'의 현실

BUILTIN

번들로 동봉. 신규 설치는 대부분 이것.

LOCAL

내가 / 에이전트가 만든·정리한 것.

HUB

레지스트리에서 설치한 것.

- 운영·정리되면 카테고리가 **통합 엄브렐러(local)** 로 묶이기도 합니다 (예: `github-workflows`).
- **개수가 줄어도 능력이 준 게 아닙니다** — 갓 설치(개별 ~90)와 정리본(통합 ~27)은 같은 능력의 다른 정리법.

스킬을 어디서 더 얻나 — 탐색 경로

신뢰 ↑ · 공식

번들 · 공식 Optional

이미 있거나 `hermes skills install official/...`

신뢰 ↓ · 커뮤니티

skills.sh · ClawHub · GitHub

남의 지침/코드 — 변동이 잦습니다.

입문 원칙: **공식·번들 중심.** 외부는 **"설치보다 inspect 먼저"**.

스킬도 공격 표면이다 — 보안

SCAN → TRUST → 설치

- hub 설치 스킬은 **스캐너**(데이터 유출·프롬프트 인젝션·파괴적 명령)를 통과해야 합니다.
- `-force` 는 경고는 넘겨도 dangerous 판정은 못 넘깁니다.
- 스킬 안에 API 키 직접 작성 금지 — `.env`·환경변수로.

비서가 스스로 매뉴얼을 쓴다

맛보기 · 절차적 기억

- 에이전트가 복잡한 일을 풀면, 그 절차를 `skill_manage` 로 **스킬로 저장**합니다.
- 단 **검토 후 채택** — 아무거나 막 쌓이지 않게.

이렇게 쌓인 스킬을 **스스로 정리**하는 게 2.11 Curator입니다.

정리 — 스킬은 비서의 업무 매뉴얼 라이브러리

① 필요할 때 **자동 로드** (progressive)

② 슬래시는 **수동 호출** = 로드 (≠ 실행)

③ **builtin · local · hub** 로 출처·신뢰 구분

④ 외부는 **inspect 먼저**

다음 → 2.8 에서 내 업무 기준으로 스킬 후보를 골라봅니다.