AIOps Course · New PKM Paradigm

새로운 지식관리 패러다임

AI 시대, 마크다운으로 다시 쓰는 PKM — **잘 정리된 지식 × AI = 무한한 가능성**

WHY	왜 지금, 새로운 패러다임인가
WHAT	마크다운 · PARA vs 제텔카스텐
HOW	OpenCode + Obsidian 조합

PKM Value Chain

지식이 가치가 되는 6단계 — 어디서 끊겼나?

한 단계라도 막히면 **전체가 무너진다**

01
소화
DIGEST
정보 홍수
읽어도 안 남음

▶

02
저장
STORE
포맷 락인
Notion에 갇힘

▶

03
진화
EVOLVE
죽은 노트
업데이트 안 됨

▶

04
연결
CONNECT
파편화
섬처럼 흩어짐

▶

05
재사용
REUSE
1회용
매번 처음부터

▶

06
AI 활용
AI-READY
형태 부재
AI가 못 읽음

우리는 ①②에만 집중해왔다 — AI 시대의 진짜 가치는 ④⑤⑥에서 나온다. 지식을 "쌓는" 것에서 **"살아 움직이게"** 만드는 것으로.

AI Augmentation Formula

인간의 잠재력 × AI 증강 = 기하급수적 성장

INPUT

입력 (배움)

책 · 아티클 · 강의
관찰 · 정보 수집

AI ENGINE

3차원 동시 증강

속도 ↑ 책 1권 → 10분 파악
양 ↑ 논문 50편 동시 리뷰
질 ↑ 무한 질문 → 깊은 통찰

OUTPUT

출력 (일)

문서 · 보고서 · 코드
설계 · 창작 · 분석

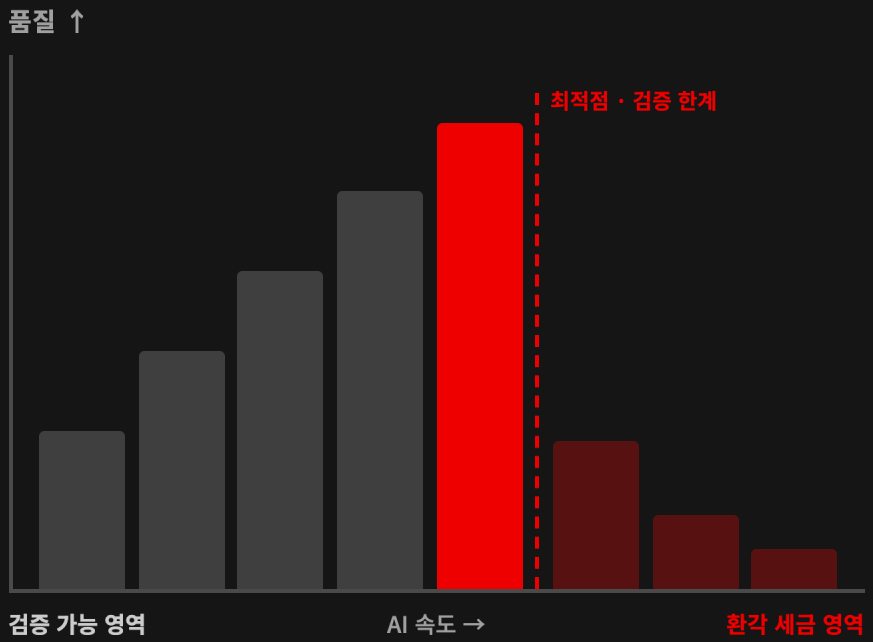
덧셈이 아닌 **곱셈** — 속도 × 양 × 질이 동시에 증가할 때 효과는 기하급수적이다

단, 전제가 있다 — 정리되지 않은 지식 × AI = **0**. 곱셈이기 때문이다

Multiplication Trap · Verification Asymmetry

검증을 못 따라가면 0이 된다

AI의 생성은 **O(1)**, 인간의 검증은 **O(n)** — 속도가 검증 대역폭을 넘는 순간 **환각 세금**이 발생한다



검증 가능

AI 속도 ≤ 검증 속도 — 비판적으로 평가할 시간 있음 → **품질 ↑, 신뢰 누적**

환각 세금 (Hallucination Tax)

검증 한계 초과 → System 1 도피 → 무비판 수용 → **품질 붕괴, 부채 누적**

빠른 AI 페이스의 **외재적 인지 부하**는 본질적 작업 난이도보다 **3배 더** 품질을 파괴한다 — arXiv 2025 (n=34)

The Real Formula

AI 속도가 아니라, 당신의 검증 대역폭이 품질의 상한이다

순진한 공식

~~Quality = Speed × Volume × Quality~~

실제 공식

Sustainable Quality = 검증 대역폭 × AI 활용도

패턴 — Centaur 모델 (Mollick · BCG, n=758)	비율	사용 방식	정확도
Centaur (켄타우로스)	**14%**	**지시된 공동 창작 — 통제된 페이스, 명확한 검증**	**최상**
Cyborg (사이보그)	60%	융합된 공동 창작 — 빠른 핑퐁, 경계 흐림	**중간**
Self-automator (자동화)	27%	AI에 위임, 검증 생략	**최하**

수렴점은 사람·작업마다 다른 **동적 평형** — AI 출력을 그냥 복붙 중이면 **너무 빠른 것**, "이게 왜 맞지?" 질문이 떠오르는 페이스가 적정

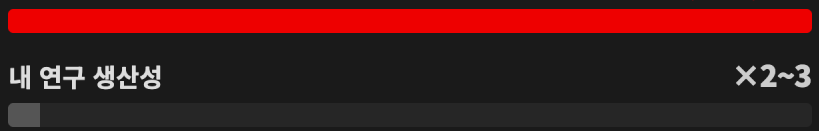
Weak Link Economics · Chad Jones (Stanford)

약한 고리의 경제학 — 검증이 당신의 몸값이 된다

챌린저호 — 3만 원짜리 고무링

250만 개 부품, 4조 원짜리 우주왕복선이 **O-ring 하나**로 발사 73초 만에 폭발 — **사슬은 가장 약한 고리만큼만 강하다** (Kremer, O-ring 이론 1993)

트랜지스터 1억 배의 역설 — 1970년대 대비



연산은 순식간 — 하지만 무슨 데이터·어떤 이론을 검증할지는 내가 결정한다. **약한 고리가 나를 제약한다**

흔해지는 것 — 가격 ↓

생성 · 예측 · 초안 · 코드 (GDP 내 컴퓨터 비중 4.5% → 3%)

희소해지는 것 — 가치 ↑

판단 · 맥락 · 신뢰 · 검증 — 영상의학과 의사, 소멸 예언(2016) 후 오히려 증가

PKM은 노트 정리법이 아니라 — **검증 대역폭에 대한 투자**다. 생성이 흔해질수록, **약한 고리를 쥔 사람이 가장 비싸진다**

Traditional PKM vs AI-Augmented PKM

같은 작업, 완전히 다른 비용 구조

작업	전통적 방식	AI 강화 방식	변화
정리	수동 분류 + 태그 부착	**자동 분류 · 태깅**	주관 → **일관성**
검색	키워드 매칭 (정확히 기억)	**의미 기반 (개념으로 찾음)**	기억력 → **이해력**
연결	직접 위키링크 생성	**관련 노트 자동 추천**	노력 → **창발성**
생성	빈 페이지에서 시작	**기존 노트 기반 합성**	0 → **누적자산**
요약	직접 발췌 · 요약	**자동 요약 + 비교**	시간 → **즉시성**

숨겨진 5번째 변화 — 노트의 정체성: 다시 보기 위한 **결과물(저장소)**에서, AI와 함께 활용하는 **원료(재료)**로

PARA · Tiago Forte — Building a Second Brain

실행 가능성으로 분류하라

P · A · R · A — **4글자가 모든 정보의 자리를 정한다**

P
프로젝트
PROJECTS

데드라인 있는 작업

지금 당장

블로그 글 마감, 이직 준비

A
영역
AREAS

지속 관리 책임

지속적으로

건강 관리, 팀 매니징

R
리소스
RESOURCES

관심사 · 참고 자료

언젠가

AI 트렌드, 요리 레시피

A
아카이브
ARCHIVES

완료 · 비활성

끝남

완료된 프로젝트

"지금 쓸 수 있는가?"로 분류	Just-in-Time 정리	도구 독립적 일관성	정기 리뷰 → Archive 이동

AI 시대 관점 — AI에게 **01-Projects/ 폴더만 가리키면 끝**. **현재 일에 집중된 컨텍스트**가 자동으로 생긴다

Zettelkasten · Niklas Luhmann

연결이 통찰을 만든다 — 90,000장 메모 상자

90,000장	70권+	400편+
30년간 쌓은 메모	출간한 책	발표한 논문

FLEETING

순간적 생각

포스트잇 같은 **임시 메모** — 며칠 내 처리

▶

LITERATURE

읽은 내용 정리

책 · 논문 요약 + **자신의 해석**

▶

PERMANENT

영구 지식

자기 언어로 재정리한 **원자 단위** — 핵심 자산

"혼자 쓰는 게 아니다 — 메모 상자와 대화하며 쓴다" · 원자성 · 자기 언어 · 밀도 높은 연결 · 폴더가 아닌 링크 — AI에겐 **그래프가 곧 지식 지도**다

PARA vs Zettelkasten

둘 다 좋은데, 철학이 다르다

기준	PARA — 행동 중심	제텔카스텐 — 사고 중심
핵심 질문	"이걸 어디서 쓰지?"	"이게 뭐랑 연결되지?"
구조	폴더 위계 (계층형)	링크 그물망 (네트워크형)
노트 단위	프로젝트 문서 (큰 단위)	원자 노트 (작은 단위)
시간 지향	현재 진행형	장기 누적형
강점 / 약점	빠른 행동 / 연결이 약함	깊은 통찰 / 정리에 시간 소요
적합한 사용자	바쁜 직장인, PM	연구자, 작가, 학습자

현실적 추천 — PARA를 뼈대로, 제텔카스텐을 살로 (하이브리드)

PARA가 **"지금 무엇을 할까"**를, 제텔카스텐이 **"이걸로 무엇을 만들까"**를 답한다 — 한 Vault에 공존하면 검증 대역폭이 자동으로 조절된다

Information Flow

정보가 흐르는 길 — 두 파이프라인의 해부학

PARA + CODE

Tiago Forte · 4단계 · 행동 중심 — 첫 질문 "이걸 어디에 넣을까?"

CAPTURE
울림 있는 것만 붙잡기

▶

ORGANIZE
PARA 분류

▶

DISTILL
Progressive Summary

▶

EXPRESS
블로그·발표·제품

Zettelkasten

Sönke Ahrens · 5단계 · 정제 중심 — 첫 질문 "이걸 어떻게 내 언어로 쓸까?"

FLEETING
즉흥 메모

▶

LITERATURE
자기 언어 요약+출처

▶

PERMANENT
원자 단위·링크 연결

▶

PROJECT
관련 카드 모음

▶

OUTPUT
논문·책·아티클

통찰 PARA는 **정보의 주소록**(Where), 제텔카스텐은 **정보의 정제 공정**(How) — 경쟁이 아니라 **다른 시간 축의 보완재**

Integrated Pipeline · 2026

AI가 들어가는 자리 — 통합 6단계 파이프라인

	STEP 1 CAPTURE	STEP 2 PROCESS	STEP 3 REFINE ★ 검증 핵심	STEP 4 CONNECT ★ 검증 핵심	STEP 5 MAINTAIN	STEP 6 EXPRESS ★★ 최고
AI 역할	웹 클리핑 음성 전사	자기 언어 재작성 자동 분류	4계층 요약 원자 단위 분해	10~20개 연결 제안 백링크 자동	고아 노트 감지 모순 식별	클러스터 → 초안 MOC 생성
인간 역할	무엇을 잡을지 선택	재작성 검토	**압축 적정선 판단**	**연결의 의미 선별**	주기적 리뷰	**최종 결정·편집**

분업 계약 검증 대역폭은 ③ ④ ⑥에 집중, 나머지는 AI에 안전하게 위임 — 각 단계는 **AI와 나의 협업 지점**이다

Lingua Franca

왜 마크다운인가? — AI 시대의 공용어

플레인 텍스트 PLAIN TEXT

그냥 텍스트 파일(.md), 어떤 에디터든 열림 — LLM이 가장 잘 다루는 포맷, **토큰 효율 최고**

포맷 락인 없음 NO LOCK-IN

마이그레이션 비용 0 — Notion API가 끊겨도, 회사가 망해도 **내 지식은 내 것**

도구 독립 TOOL AGNOSTIC

Obsidian·VSCode·Vim·터미널 어디서든 — AI 에이전트가 **직접 읽고 쓰기 쉬움**

미래 호환성 FUTURE PROOF

Plain Text 1972 → Markdown 2004, 50년+ 검증 — **누적 자산이 사라지지 않음**

전제 조건 AI가 노트를 **직접 읽고 써야** 검증 루프가 빠르게 돈다 — 마크다운은 AI 시대 PKM의 OS-level 결정

GFM vs Obsidian Markdown

같은 마크다운인데, 표현력이 다르다

문법	GFM (일반)	Obsidian Markdown
링크	`[페이지](페이지.md)`	`[[페이지]]` — 양방향 링크·자동 백링크
임베드	`![](image.png)` 이미지만	`![[노트]]` — 다른 노트도 임베드
Callout	`> 인용문` 단순 인용	`> [!tip] 제목` — 의미가 있는 강조
Properties	모름	`tags: [ai, pkm]` — 구조화된 메타데이터

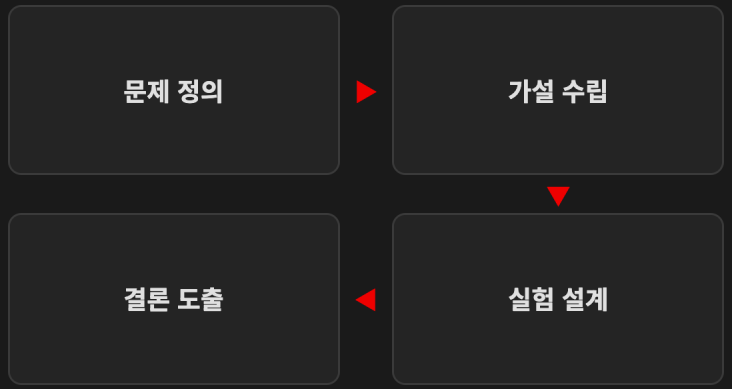
일반 마크다운은 AI가 **텍스트만** 보고, Obsidian Markdown은 **연결·메타·구조**까지 본다 — **풍부한 문법 = 풍부한 컨텍스트 = 더 정확한 AI 응답**

Beyond Markdown

Canvas & Bases — 평면 텍스트에서 입체적 사고로

JSON Canvas .canvas

시각적 사고 — 마인드맵·플로차트·아키텍처



Open Spec (jsoncanvas.org) · 카드에 노트 임베드 · JSON이라 **LLM이 읽고 생성 가능**

Obsidian Bases .base

노트의 데이터베이스 뷰 — 태그·속성으로 동적 필터

```
filters: tag.contains("ai") and mtime > 2026-01-01
```

file.name	status	tag
AI-agent-notes	draft	ai
pkm-paradigm	done	ai, pkm

테이블·카드·칸반 뷰 · **Notion DB의 오픈소스 대안** — 데이터는 마크다운 그대로

진짜 천재성 Canvas의 카드도 Base의 행도 결국 **.md를 가리키는 포인터** — AI는 **단 하나의 truth source**만 읽으면 된다

3 Levels of AI Usage

AI 활용의 3단계 진화 — 얼음에서 눈덩이로

	LEVEL 1 **채팅**	LEVEL 2 **IDE 내장**	LEVEL 3 **CLI 에이전트**
대표 도구	ChatGPT, Gemini	Copilot, Cursor	**OpenCode, Claude Code**
비유	얼음조각 모으기	내가 굴리는 눈덩이	**AI가 굴리는 눈덩이**
작업 단위	대화 한 턴	한 줄, 한 함수	**프로젝트 전체**
파일 접근	없음	열린 파일만	**전체 탐색·생성·수정**
병목	복붙 속도	타이핑 속도	**판단·방향 설정**

Level 3 = **Vault에 AI가 직접 들어온다** — 인간은 방향만, 실행은 AI가. 단, 검증을 못 따라가면 **환각 세금**이 눈덩이처럼 커진다

Toolchain Synergy

OpenCode + Oh-My-OpenCode + Obsidian — 3원조 시너지

OpenCode

CLI 에이전트

- **AI 대화** 인터페이스
- 파일 **read / write / search**
- **MCP** 외부 도구 연동

PLUGIN **Oh-My-OpenCode**

Sisyphus·Oracle·Explore 멀티 에이전트 · **Skill 시스템** · Todo 관리 — 한국 개발자 제작, 27.9k+ 스타

◀▶

같은 **마크다운 파일**을 읽고 쓴다

Obsidian

Vault · 지식 저장소

- **마크다운 노트** 저장소
- **위키링크 그래프** — 연결 시각화
- **Canvas · Bases** 렌더링
- 사람의 **편집·리뷰** 인터페이스

같은 파일시스템 — Vault = AI의 작업 공간

시너지 **데이터는 한 곳, 인터페이스는 여러 개** — 사람이 Obsidian에서 고치면 AI가 즉시 인식, AI가 만들면 그래프에 자동 반영

Agent Ecosystem

에이전트 생태계 — 그리스 신화로 분업하라

코어 — 비용 중간

Sisyphus
메인 오케스트레이터 · Todo · 위임

Atlas
계획 실행 오케스트레이터

Prometheus
전략적 플래너 · 요구사항 인터뷰

자문 — 비용 높음

Oracle
아키텍처 결정 · 어려운 디버깅

Metis
사전 계획 컨설턴트 · 모호함 분석

Momus
계획 검토자 · 실행 가능성 검증

실행 — 비용 낮음~중간

Hephaestus
자율 딥 워커 · 목표 지향 실행

Explore
빠른 노트·코드 검색 (병렬 가능)

Librarian
외부 문서·GitHub 예제 검색

"AI 노트 다 찾아줘" → **Explore**

"라이브러리 사용법 정리" → **Librarian**

"Vault 구조 개선?" → **Oracle**

"이번 주 학습 계획" → **Prometheus**

우회 전략 멀티 에이전트 = **검증 대역폭 우회** — AI끼리 서로 검증하고(Prometheus → Momus → Atlas), 인간은 최종 결정만

kepano / obsidian-skills · Obsidian CEO 제작

Obsidian Skills — AI에게 위키링크를 가르쳐라

obsidian-markdown
위키링크 [[ ]] · 임베드 ![[ ]] · Callout · Properties

obsidian-bases
.base 파일 — 뷰·필터·수식·요약

json-canvas
.canvas 파일 — 노드·엣지·그룹·연결

obsidian-cli
CLI로 Vault 조작, 플러그인·테마 개발

defuddle
웹 페이지 → 클린 마크다운 추출 (토큰 절약)

스킬 유무의 차이

스킬 없이
[페이지](페이지.md) — 일반 링크, 백링크 없음

스킬 설치 후
[[페이지]] — 양방향 링크 · 정확한 frontmatter · .canvas / .base 자동 생성

```
> /plugin marketplace add kepano/obsidian-skills
> /plugin install obsidian@obsidian-skills
```

일관성 Skill = **AGENTS.md의 모듈화** — 컨벤션을 한 번 정의하면 AI가 영원히 따른다, 매번 가르칠 필요가 없다

Wrap-up

핵심 정리 — 새로운 패러다임의 6가지 축

01	노트는 결과물이 아니라 **재료**다	누적 자산
02	Quality는 곱셈의 항이 아니라 **시스템 제약**이다	검증 대역폭
03	방법론은 양자택일이 아니라 **하이브리드**다	PARA × 제텔카스텐
04	마크다운은 "어쩔 수 없이"가 아니라 **AI 시대의 공용어**다	Plain Text 영구성
05	AI는 혼자가 아니라 **에이전트 팀**으로 일한다	Sisyphus + Oracle + Explore
06	AI가 대체하는 건 직업이 아니라 **과업**이다 — 약한 고리를 쥔 사람이 비싸진다	Weak Link 경제학

"AI가 엔지니어를 대체하는 게 아니라, AI를 쓰는 엔지니어가 안 쓰는 엔지니어를 대체한다."

Atul Gawande (하버드 의대) — "AI를 쓰는 의사가 안 쓰는 의사를 대체할 것이다"에서

잘 정리된 지식(.md) × AI 에이전트 팀 × 검증 페이스 = 새로운 PKM 패러다임

Next Steps

강의는 끝났지만, 패러다임 이동은 시작이다

STEP 1 · 30분

Vault 만들기

- Obsidian 다운로드 → **Vault 생성**
- 폴더 구조: **PARA + 제텔카스텐 하이브리드** (Slide 10 참고)

STEP 2 · 1시간

에이전트 설치

- `opencode.ai/install` 스크립트 실행
- `oh-my-opencode` + `obsidian-skills` 플러그인 설치
- Vault에서 **첫 노트를 AI에게** 만들게 하기

STEP 3 · 일주일

검증 페이스 찾기

- 매일 1번씩 **AI에게 노트 작성 의뢰**
- "그냥 복붙"한 순간 기록 = **너무 빠른 페이스**
- 자기만의 **수렴점** 발견

NEXT agent-md Course 다음 모듈 — **AGENTS.md 완벽 가이드**: 작고 강력한 AI 설정 만들기