KOICA 해외사무소 · AI & DATA

AI·데이터 활용,
해외사무소의 일을 다시 설계하다

Claude Code · MCP · LLM Wiki · Skills · Plugins · Web · GitHub

1부

오늘의 흐름

01 시작
Claude Code · Codex

02 연결
MCP 도구 · 업무 흐름

03 변환
바이너리 → 텍스트

04 지식
Brain dumping · LLM Wiki

05 재사용
Skill · Plugin

06 웹앱 선택
계획 모드 · 배포

07 축적
GitHub · 원격 확인

08 사례
KOICA 동료 프로젝트

09 트렌드
계속 따라가는 방법

실습 환경은 macOS와 Windows를 함께 지원한다

macOS

Claude Code + Terminal

```
python3 -m http.server 8000
```

Windows

Claude Code + PowerShell / Git Bash

Git for Windows

```
py -m http.server 8000
```

공통 준비

Node.js · 최신 브라우저 · 프로젝트 폴더

공백이 있는 경로는 따옴표로 감싼다

6부 · 선택 Cloudflare Wrangler는 macOS 13.5+ · Windows 11에서 실습합니다. Windows 10은 시연만 봅니다.

PART 01

AI를 대화창 밖으로 꺼낸다

Claude Code와 Codex로 파일을 남기는 일부터

01

좋은 답변보다 좋은 흐름이 오래 남는다

질문 한 번 → 답변 한 번 **업무 한 번**

자료 → 근거 → 산출물 → 공유 → 개선 **업무가 이어지는 흐름**

Claude Code로 시작하고, ChatGPT(Codex)도 함께 쓴다

Claude Code는

- 프로젝트 폴더를 읽고
- 파일을 만들고 고치며
- 명령을 실행하고
- 외부 도구를 연결한다

ChatGPT의 Codex에서도 같은 방식으로 일할 수 있다

이 강의자료도 과거 프로젝트를 분해·재조립해 만들었다

lunch-time-claude-camp

과거 교육 자료

DPGA 자료

개인 프로젝트 기록

→

AI AGENT

읽기 · 분해

선별 · 재연결

→

현재 프로젝트

KOICA AI·데이터 교육자료

다음 작업의 재료로 남는다

AI와 함께 일한 결과가 **다음 프로젝트의 재료**가 됩니다.

PART 02

도구와 원천을 연결한다

말 잘하는 AI를 근거를 확인하는 업무 도구로

02

AI에게 손을 달아주는 표준, MCP

AI에게 **손을 달아주는** 공통 연결 단자

첫 번째 연결: KOICA 규정을 직접 조회한다

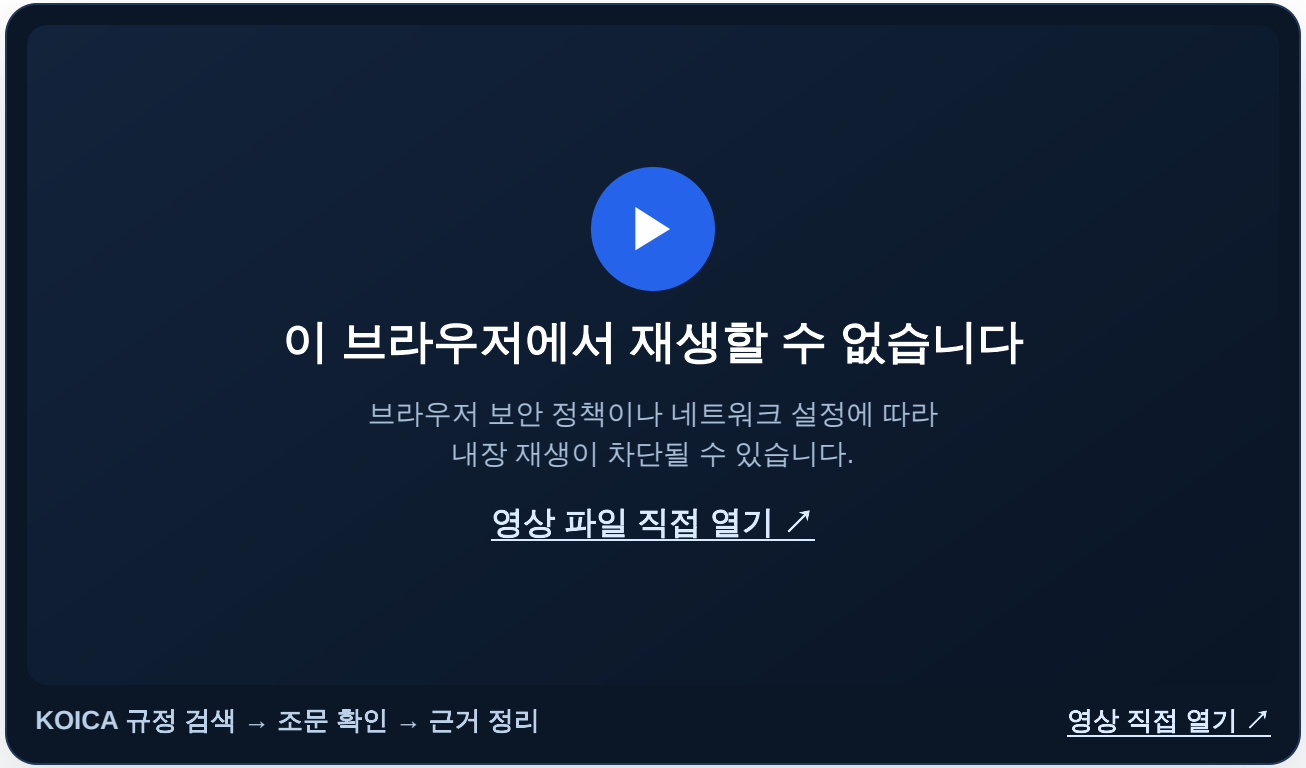
koica-reg

“이 규정이 실제로 존재하는가?”

“정확히 몇 조인가?”

“관련 규정은 어디로 이어지는가?”

도구 안내 · amnotyoung/koica-reg-mcp ↗



KOICA 규정 검색 → 조문 확인 → 근거 정리

영상 직접 열기 ↗

두 번째 연결: 내부 규정 밖의 법령을 확인한다

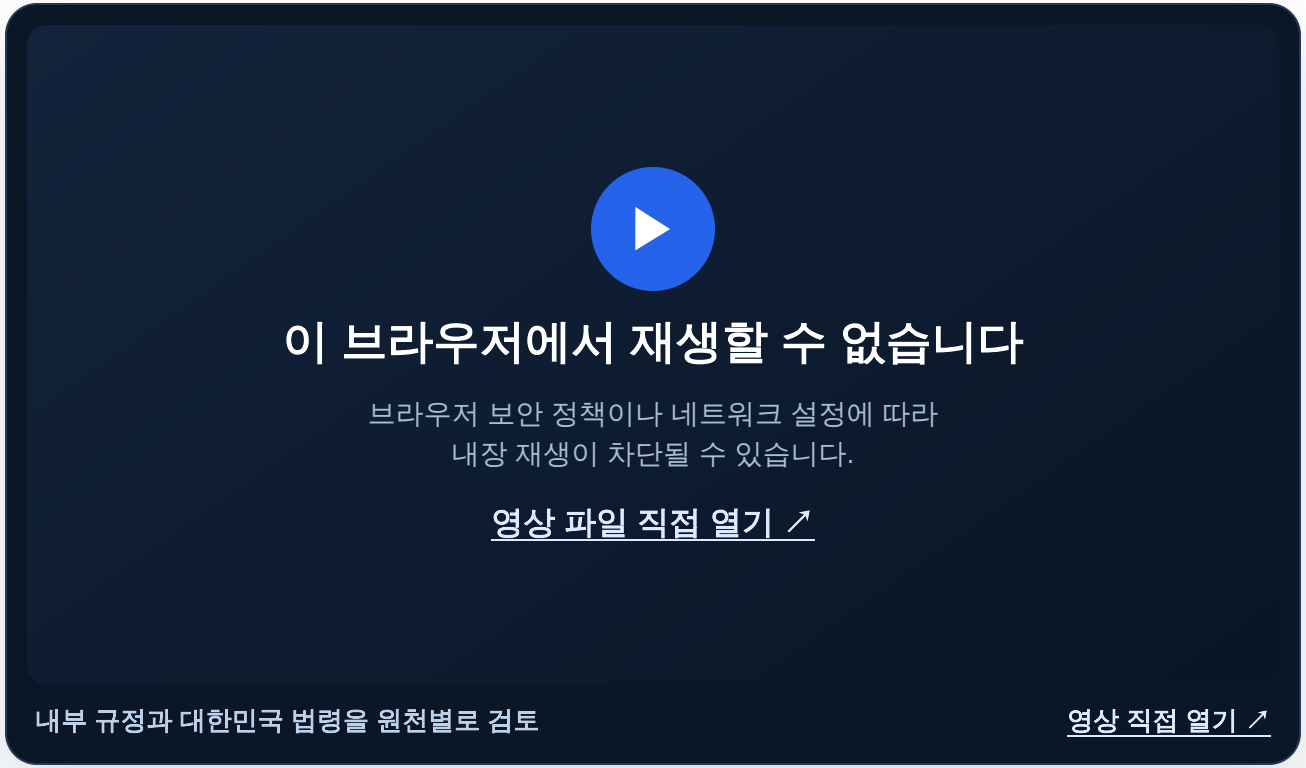
korean-law

KOICA 규정과 한국 법령은

서로 다른 원천이다

필요하면 두 도구를 함께 사용한다

도구 안내 · chrisryugj/korean-law-mcp ↗



내부 규정과 대한민국 법령을 원천별로 검토

영상 직접 열기 ↗

세 번째 연결: 읽기 어려운 문서를 텍스트로 바꾼다

kordoc

HWP · HWPX · PDF · Office

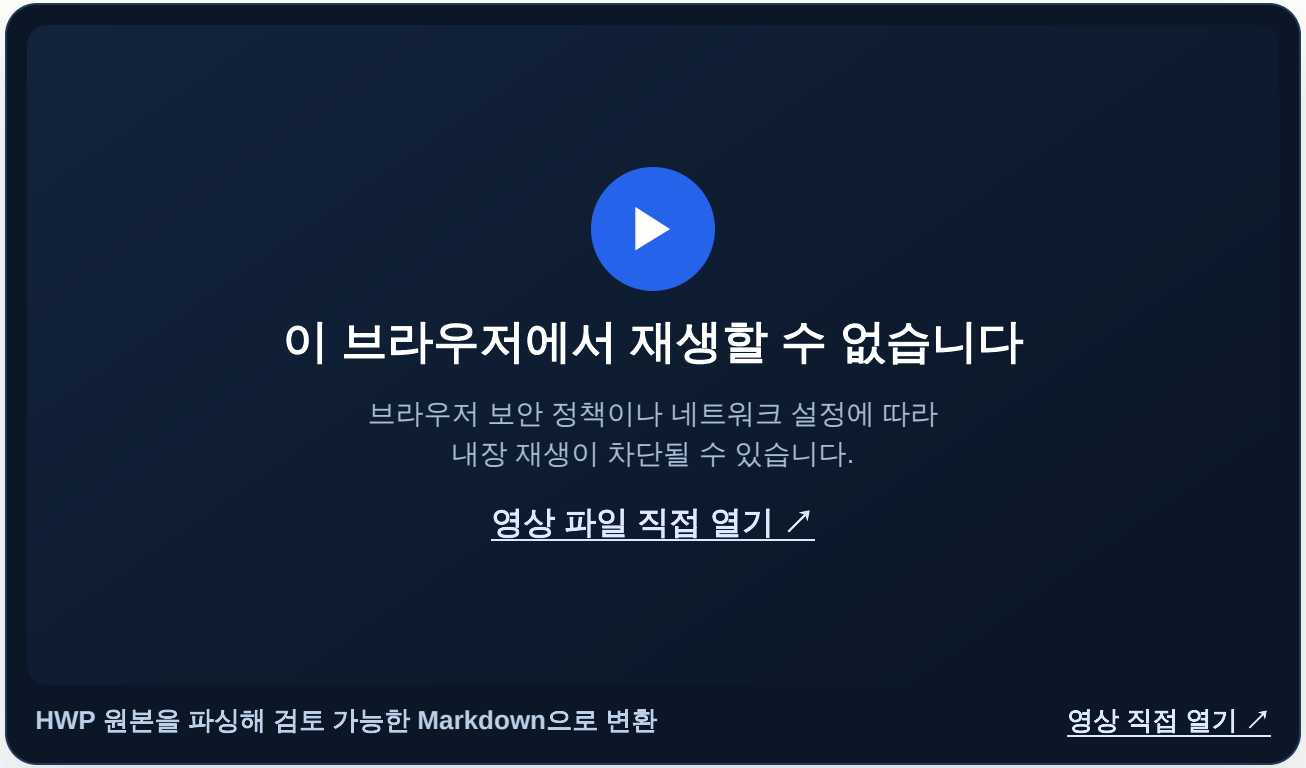
↓

파싱

↓

Markdown

도구 안내 · **chrisryugj/kordoc** ↗



HWP 원본을 파싱해 검토 가능한 Markdown으로 변환

영상 직접 열기 ↗

마지막 연결: 회의도 검색 가능한 자료가 된다

Tiro

회의 음성

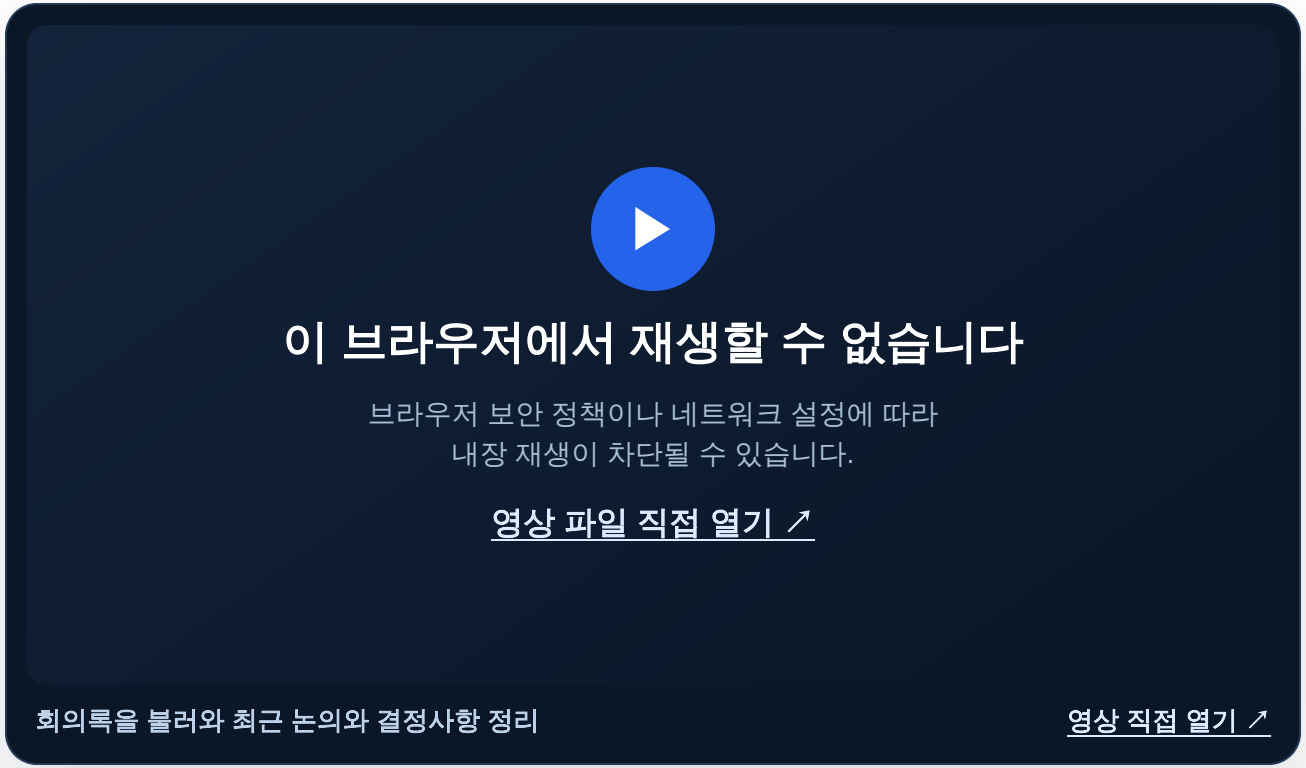
↓

회의록.md · 요약 · 화자별 발화

↓

Claude Code의 컨텍스트

Tiro MCP 공식 안내 ↗



네 도구가 하나의 업무로 연결된다

01 · 회의
Tiro 회의록.md

02 · 기존 문서
HWP → kordoc 기존문서.md

→

03 · 연결
회의록과 기존 문서
관련 내용을 함께 읽는다

→

04 · 검토
koica-reg + korean-law
규정·법령 근거 확인

→

05 · 산출물
MD · CSV · 근거표

도구가 많아서가 아니라 흐름이 이어져서 강하다

도구 **4**개를 아는 것 ≠ 업무가 자동으로 좋아지는 것

입력 → 변환 → 검증 → 산출물 순서가 있어야 한다

PART 03

파일을 AI가 읽는 형태로 바꾼다

보이는 원본을 검산 가능한 텍스트로

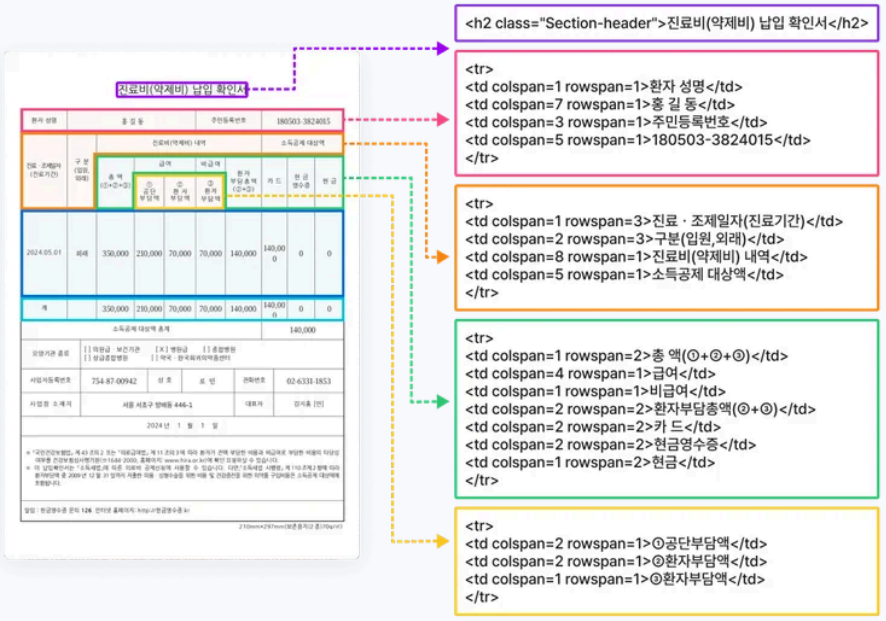
03

파일이 열린다고 AI가 잘 읽는 것은 아니다

사람에게는 **한 장의 문서**

AI에게는 **해석해야 할 구조**

> 파일을 열 수 있다 ≠ 내용을 정확히 이해했다



보이는 문서 한 장이 내부에서는 태그·행·병합 셀 구조로 바뀐다.

사용자 제공 URL · rights unverified · internal only

바이너리 파일에는 번역 단계가 숨어 있다

HWP

HWPX

PDF

Office

kordoc

HWP → MD 파싱

Markdown + 구조

사람이 검산할 수 있는 형태

파싱이 틀리면 답변도 자신 있게 틀릴 수 있습니다.

목적에 맞는 텍스트 형식이 있다

형식	잘하는 일
`.md`	문서, 지침, Wiki, 보고서
`.csv`	표, 목록, 집계, 검산
`.txt`	원문, 로그, 단순 기록
`.html`	브라우저에서 공유하는 결과물

텍스트는 변경점을 비교하고 되돌릴 수 있다

텍스트 파일은

- 바로 검색할 수 있고
- **Git diff**로 변경 전후를 비교할 수 있고
- 필요하면 이전 상태로 되돌릴 수 있고
- 여러 결과물로 바꿀 수 있다

텍스트 우선은 최종 형식이 아니라 작업 원칙이다

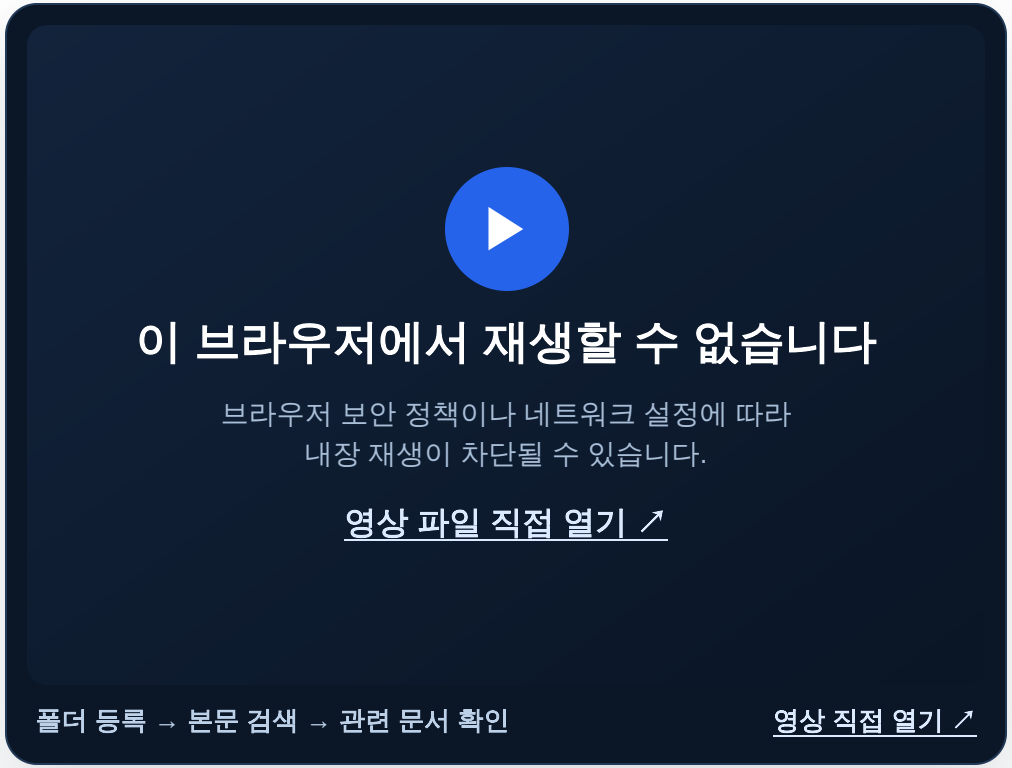
1. 원본은 보존한다
2. 작업용 텍스트를 만든다
3. 변환 결과를 검산한다
4. 텍스트에서 반복 작업을 수행한다
5. 필요하면 Office·PDF·웹으로 내보낸다

Docufinder는 파일 이름 대신 본문으로 찾는다

Docufinder · Anything

- 폴더를 등록하면 문서 본문을 자동으로 인덱싱
- HWP · PDF · Office · 이미지까지 검색
- 파일 이름을 몰라도 내용으로 찾기
- 온디바이스 AI로 연결하면 완전 로컬 검색 도구로 사용

chrisryugj/Docufinder ↗



PART 04

쌓인 자료를 관계 있는 지식으로 만든다

Brain dumping에서 LLM Wiki로

04

먼저 쌓고, AI와 함께 정리한다

Brain dumping

완벽하게 분류한 뒤 기록하지 않는다

생각 · 링크 · 문서 · 메모를 먼저 남긴다

→

LLM

Parsing
Indexing

나중에 정리한다

사용자는 **쌓고**, 정리와 색인은 **LLM이 맡습니다.**

실습: 자료 폴더를 LLM Wiki로 바꾼다

01 · USER
Raw data 폴더
자료를 계속 한곳에 넣는다

→

02 · LLM
Parsing · Indexing
페이지·관계 생성과 유지관리

→

03 · OPTIONAL
Obsidian
Markdown Wiki를 직접 탐색

> **“이 자료들로 LLM Wiki를 만들고 관리해줘.”**

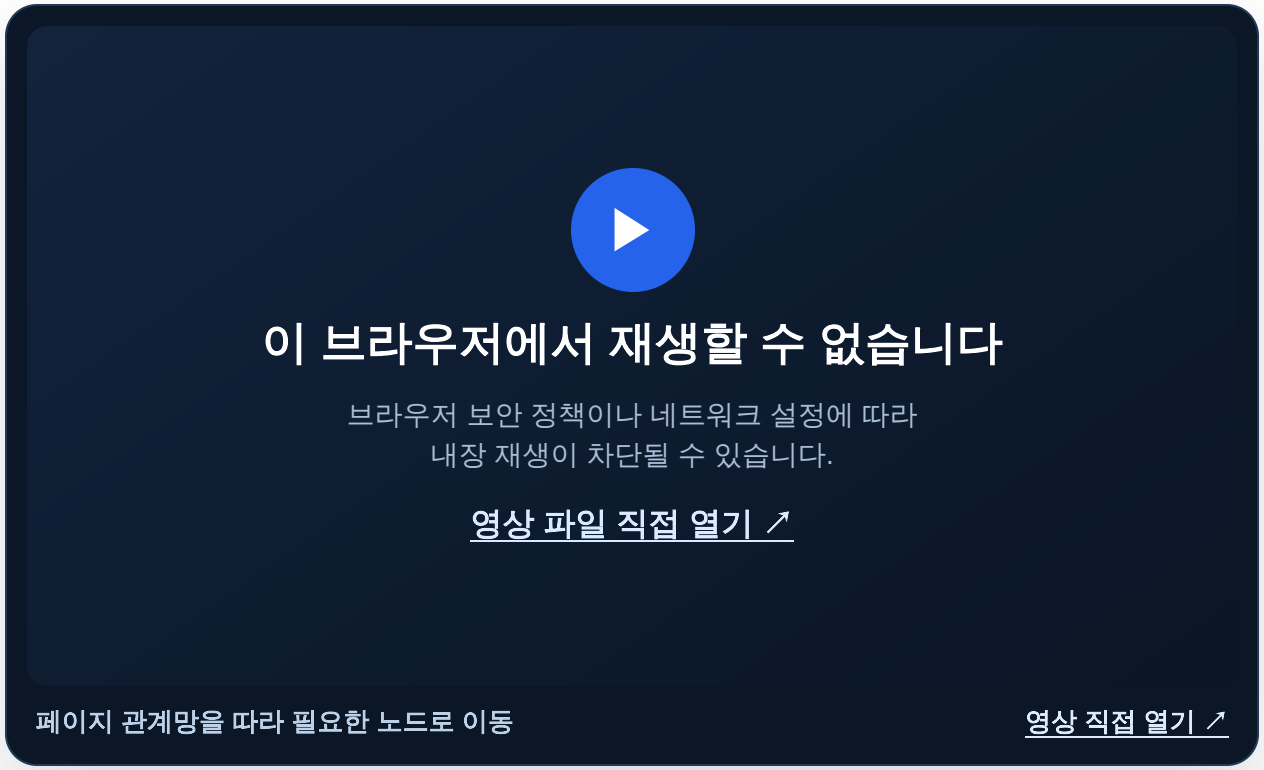
사용자는 **raw data**를 넣고, **LLM**이 **parsing·indexing·관리**를 맡습니다.

LLM Wiki는 검색·원문·연결을 한 흐름으로 탐색한다

LLM Wiki의 기본 흐름

1. **검색** 관련 페이지를 찾는다
2. **원문** 페이지 내용을 확인한다
3. **언급** 연결된 노트와 문서를 추적한다
4. **그래프** 페이지 사이 관계망을 따라간다

AI는 질문에 필요한 깊이만큼 단계적으로 조회한다



페이지 관계망을 따라 필요한 노드로 이동

영상 직접 열기 ↗

AI가 모르는 관련성을 사용자는 알고 있다

세네갈 농업사업

비슷한 시행착오

담당자 교체와 인수인계 실패

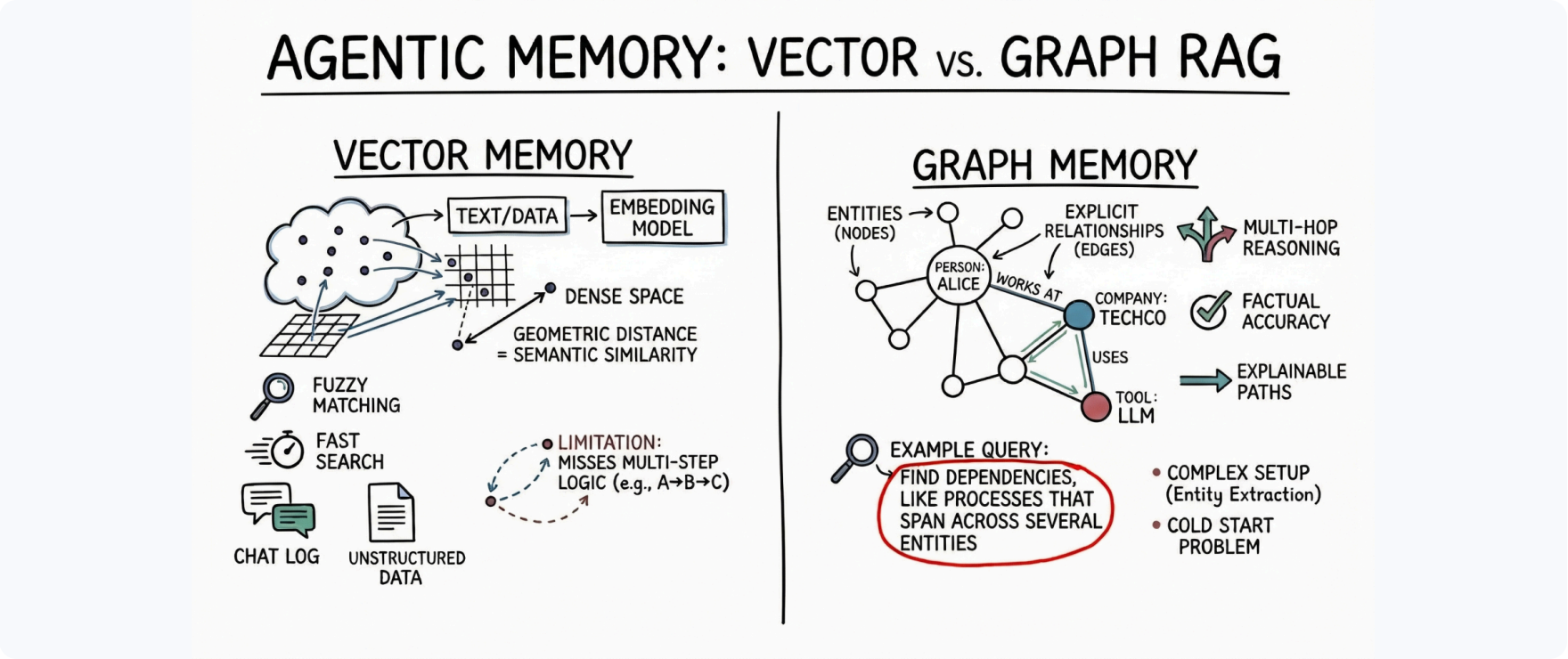
우간다 청년고용사업

AI가 모르는 관련성을 **사용자는 알고 있습니다.**

RAG는 찾고, Wiki는 관계를 고칠 수 있다

RAG	LLM Wiki
질문과 가까운 문서를 검색	페이지와 링크가 눈에 보임
관련성을 시스템이 계산	사용자가 관련성을 추가·수정
검색 결과가 질의 때 나타남	관계가 지속적인 지식으로 남음

GraphRAG는 문서가 아니라 관계망도 따라간다



Vector는 가까운 문서를 찾고, Graph는 명시된 관계의 경로를 따라간다.

사용자 제공 URL · rights unverified · internal only

중요하지만 귀찮은 Lint와 Bookkeeping은 LLM에게 맡긴다

LLM이 정기적으로 찾아낼 것

- 고립된 페이지
- 깨진 링크
- 출처 없는 주장
- 중복 페이지
- 이유 없는 관계
- 오래된 정보

LLM Wiki의 가치는 정리 노동을 계속 대신하는 데 있다

Raw data → Parsing → Indexing → 관계 연결 → Lint → 보고

사용자는 원자료를 넣고 판단합니다. **정리·색인·점검은 LLM이 반복합니다.**

PART 05

반복은 Skill로, 공유는 Plugin으로

한 번 잘한 일을 동료도 재현하게

05

같은 질문에 매번 설명을 다시 붙이지 않는다

관련 페이지를 찾고 원문을 확인하고 근거를 붙이고 사용자 관계와 AI 관계를 구분하고…

→

/wiki-answer

프롬프트를 반복하지 말고 **일하는 절차를 저장합니다.**

스킬은 Claude에게 주는 업무 매뉴얼이다

Skill

- 언제 사용해야 하는지
- 어떤 순서로 일하는지
- 무엇을 참고해야 하는지
- 어떤 형식으로 답하는지

AI 에이전트에게 스킬을 만들게 한다

오늘 배울 것

1. 반복해서 잘 된 업무를 고른다
2. AI 에이전트에게 “이 절차를 스킬로 만들어줘”라고 요청한다
3. 다른 질문으로 실행해 검증한다
4. 규칙과 예시를 보완한다

Wiki 답변 스킬은 이 과정을 배우기 위한 예시다

내장 skill-creator로 만들고, 검증하며 버전을 쌓는다

01
잘 된 대화

→

02
skill-creator
Anthropic 제공 · 제작·평가 안내

→

03
다른 질문
검증

→

04
규칙 · 예시
보완

→

05
Git
버전 기록

검증과 보완을 반복하면 **버전관리의 필요성**이 자연스럽게 생깁니다.

스킬은 다른 질문에서도 같은 규율을 지켜야 한다

검증 질문

- 정확한 답이 있는 질문
- 여러 페이지를 연결해야 하는 질문
- 근거가 부족한 질문
- 사용자가 만든 관계가 필요한 질문

스킬이 일하는 방법이라면, 플러그인은 설치 단위다

방법

Skill

Claude가 일하는 절차

전달

Plugin

그 능력을 묶어 설치·공유

사용자 제공 exact asset · 원출처 미상 · 내부 라이브 교육용 예외 검토

Skill은 방법, Plugin은 전달.

플러그인에는 필요한 구성요소만 넣는다

`skills/`	이번 실습	반복 가능한 Wiki 답변 규칙
`agents/`	선택	전문 역할이 필요할 때만
`hooks/`	선택	이벤트 자동화가 필요할 때만
`.mcp.json`	선택	외부 도구 연결이 필요할 때만

모든 플러그인에 Agent와 Hook이 있는 것은 아닙니다.

플러그인은 동료가 설치할 수 있어야 완성된다

만든 사람의 컴퓨터에서만 작동 → 개인 자동화

동료가 설치하고 같은 결과를 재현 → 공유 가능한 역량

MCP·스킬·플러그인은 서로 다른 문제를 푼다

질문	선택
외부 데이터와 도구를 어떻게 연결할까?	MCP
Claude가 어떤 절차로 일해야 할까?	Skill
이 능력을 어떻게 묶어 동료에게 전달할까?	Plugin

PART 06 · OPTIONAL

선택 실습 · 계획부터 웹앱 배포까지

현장 분위기와 남은 시간에 따라 진행하거나 건너뛴다

06

Claude Code가 없는 동료에게는 어떻게 공유할까?

지금까지 만든 **Wiki**와 스킬을 **브라우저에서도 사용할 수 있게 하자**

계획 모델과 실행 모델을 나눈다

CLAUDE CODE **Claude**	PLAN **Opus**	→	BUILD **Sonnet**
CHATGPT CODEX **ChatGPT**	PLAN **sol**	→	BUILD **terra**

계획은 강한 추론에, 반복 실행은 빠른 모델에 맡깁니다.

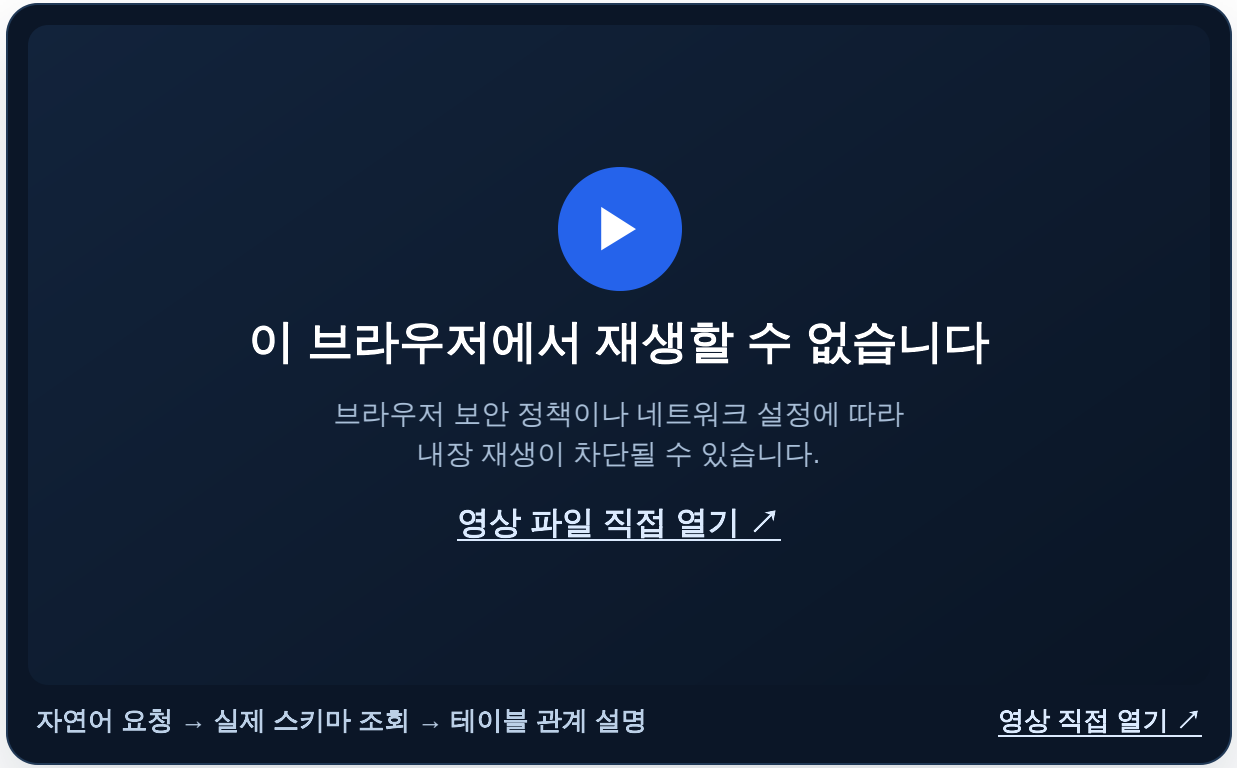
웹앱은 세 층으로 이해하면 된다

프론트엔드	사용자가 보고 누르는 화면
백엔드	요청을 처리하는 로직
데이터베이스	기억해야 할 정보

데이터베이스도 자연어로 관리할 수 있다

Supabase MCP에 물어본다

- “현재 테이블 구조를 설명해줘.”
- “Wiki 페이지와 관계를 저장할 구조를 제안해줘.”
- “예제 데이터를 넣고 국가별 건수를 집계해줘.”



우리가 만들 웹앱의 최소 구조

브라우저
검색 · 질문 · 관계 추가

↕

Cloudflare Pages
웹 화면과 앱 로직

↕

Supabase
pages · relations · sources

Cloudflare Pages에 완성 폴더를 직접 올린다

처음 한 번

로그인

npx wrangler login

→

배포

폴더 배포

Direct Upload · 없으면 질문 후 생성

→

결과

pages.dev

브라우저 공유

```
npx wrangler pages deploy <폴더>
```

개발이 끝나면 마지막으로 이 문장을 입력한다

> “이 프로젝트의 보안 취약성을 검토해줘.
> 중요도순으로 설명하고, 수정안을 제시해줘.”

수정 후 다시 묻는다

> “남아 있는 취약성을 다시 검토해줘.”

PART 07

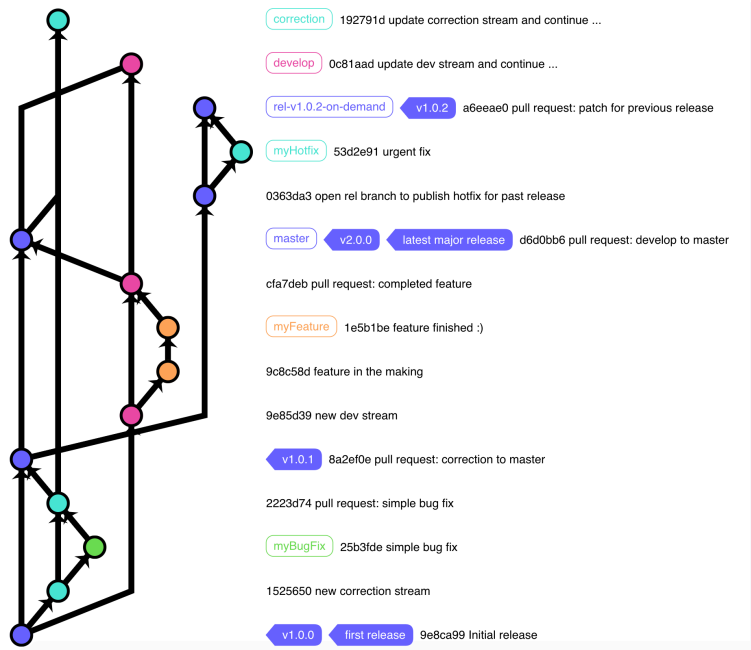
GitHub에 축적하고 원격으로 확인한다

변경을 기록하고, 자리를 비워도 상태를 본다

07

Git은 폴더의 타임머신이다

- 무엇이 바뀌었는지 본다
- 중요한 순간을 저장한다
- 이전 상태로 돌아간다
- 실험을 두려워하지 않는다



커밋과 브랜치가 남기는 변경의 지도

사용자 제공 URL · rights unverified · internal only

GitHub는 저장소를 협업 공간으로 바꾼다

내 컴퓨터

Git

변경 이력 · 되돌리기 · 실험

함께 쓰는 공간

GitHub

공유 · Issue · PR · Review · 배포

내 GitHub 저장소는 업무의 기억과 공유점이 된다

내 저장소가 남기는 것

Files	문서 · 데이터 · 교육자료
README	목적과 사용법
Commits	변경 이력과 중요한 버전
Issues	할 일 · 오류 · 아이디어

AI에게 맡길 수 있는 일

- “README를 최신 상태로 고쳐줘.”
- “열린 Issue를 우선순위별로 정리해줘.”
- “변경점을 검토하고 커밋해줘.”
- “교육자료를 웹페이지로 배포해줘.”

개발 코드가 없어도 **업무의 기억과 공유점**이 됩니다.

Issue는 문제를 기록하고, PR은 변경을 검토한다

Issue

해야 할 일, 오류, 아이디어, 질문

Pull Request

실제 변경점, 검토, 승인, 반영

승부는 자리에 있을 때보다 비웠을 때 난다

자리를 비우기 전에

- 완료 조건
- 자동 검증
- 중단 조건
- 승인 지점

목표 → 실행 → 검증 → 수정

최근의 승부처는 일할 때의 생산성보다 **자는 동안에도 일이 진행되는가**입니다.

하네스 없이 밤새 돌리면 실패를 자동화한다

01 · 시작

AI를 밤새 실행

02 · 하네스 없음

완료 조건 · 자동 검증 생략

03 · 남 탓

“역시 AI는 못 믿어”

Corentin Penloup — Baton Roue 원작 · 내부 라이브 교육용 예외 검토

MCP + Skill + Plugin + 폴더 구조 + 완료 조건 Harness

하네스 없이 돌리고, 실패는 AI 탓.

Codex Remote는 데스크톱과 모바일을 QR로 연결한다

1. Mac·Windows의 ChatGPT 데스크톱 앱을 최신화한다
2. 호스트에서 **Set up Remote**를 열어 QR 코드를 띄운다
3. ChatGPT 모바일 앱으로 스캔하고 계정·워크스페이스·MFA를 확인한다
4. **Settings → Connections**에서 연결 기기와 절전 설정을 확인한다

공식 안내 · **Remote connections**

PART 08

KOICA 동료의 실제 프로젝트

작은 불편함이 도구와 공공자산으로 이어진다

08

모두 작은 불편함과 개인적 호기심에서 시작했다

- 개발협력 동향을 더 쉽게 찾고 싶었다
- 사업 위치를 지도에서 보고 싶었다
- 규정을 정확히 확인하고 싶었다
- 협력국의 조달 절차를 미리 알고 싶었다
- 평가의 근거를 더 잘 남기고 싶었다
- 개발협력의 딜레마를 체험하게 하고 싶었다
- 이것들을 하나씩 붙이지 않고 한 번에 쓰고 싶었다

Explore와 Map: 흩어진 자료를 맥락과 공간으로 연결

DevCoop KG

- 국가별 동향 Wiki
- 기관·사업·주제 관계망
- LLM 질의

웹에서 열기 ↗

MCP 서버
`https://devcoop-trends-wiki.pages.dev`

ODA Map Lab

- 해외 위치 시각화
- 좌표와 정보 검토
- 사용자의 보정 제안

웹에서 열기 ↗

MCP 서버
`https://oda-map-lab.pages.dev/mcp`

Verify와 Evaluate: 규정·조달·평가에 근거를 남긴다

KOICA Regulation MCP

- 규정 검색
- 조문 조회
- 인용 검증

GitHub에서 열기 ↗

그 나라 조달은 어떻게 할까?

- 44개 협력국 + KOICA
- 입찰 · 거버넌스 · 사업형성
- 조문 · 기한 · 보증 · 이의 대조

웹에서 열기 ↗

DevEval Agents

- 평가 기준 적용
- 근거 게이트
- 사람의 최종 판단

GitHub에서 열기 ↗

Learn과 Package: 체험으로 배우고 설치로 전달한다

Aid World

- 개발협력 현장의 선택과 딜레마
- 교육용 내러티브 게임

GitHub에서 열기 ↗

ODA Intelligence Plugin

- Skill 3개 + 커넥터 1개로 설치
- 네 백엔드를 읽기 전용 도구 30개로
- Claude · Codex · ChatGPT

GitHub에서 열기 ↗

MCP 서버

`https://oda-mcp.fly.dev/oda-intelligence/v2/mcp`

개인 프로젝트도 디지털공공재의 씨앗이 될 수 있다

공개했다고 바로 **DPG**는 아니다

9개 지표를 점검하며 DPG 후보에 가까워진다

- **SDG** 관련성
- 공개 라이선스와 명확한 소유권
- 플랫폼 독립성과 문서화
- 비개인정보 추출과 관련 법 준수
- 공개 표준·개인정보·보안
- 불법·유해 콘텐츠와 괴롭힘 방지

PART 09

트렌드를 따라가는 계정을 찾는다

최신 트렌드와 함의, 유즈케이스를 소개하는 계정을 따라간다

09

트렌드는 뉴스보다 사용 장면에서 먼저 보인다

META · TEXT-FIRST SOCIAL PLATFORM

Threads

텍스트 업데이트와 공개 대화를 중심으로
사용 사례·신기능·현장 반응이 빠르게 흐릅니다.

threads.net ↗

추천 계정

@choi.openai	AI 트렌드와 함의
@chris_gomdori	개발·에이전트 유즈케이스
@amnotyoung.k	강사 계정 · 개인 프로젝트와 실험

뉴스가 되기 전의 **사용 장면과 해석**을 따라갑니다.

불편함을 들여다보는 순간부터 시작된다

불편함에 대해 **LLM과 대화를 시작하고**

반복을 **Skill**로 묶고

결과를 함께 쓰는 **자산**으로 축적합니다

> 오늘 하나의 프로젝트를 시작하세요.