ACTIGENCE

AI 업무 활용 과정 · 2회차

코딩 없이,
내 업무 도구를, 내가 직접

클로드 코드를 깊게 쓰고, 말로 만든 도구를 주소와 데이터까지 올립니다

PART 01 · 깊게

클로드 코드를 깊게 씁니다

모델 · 사고 강도 · 서브에이전트 · 울트라코드

모델 · 사고 강도 — 기본값은 절반만 쓰는 것

같은 클로드라도 설정 두 개로 결과가 달라집니다.

설정	기본값	바꿀 값	무엇이 달라지나
1. 모델	플랜마다 다름	더 깊이 생각하는 상위 모델	데이터 정합성·팩트체크가 필요한 일 = '실무'
2. 사고 강도(effort)	모델마다 다름	높음 ~ 최대(max)	더 오래 생각해서 놓치는 조건·예외가 줄어듭니다

문장 다듬기 · 요약 · 짧은 질문	크게 상관없지만 그래도 상위 모델 이상
보고서 · 기획서 · 데이터 분석	상위 모델 + 사고 강도 높음 ~ 최대
결재 문서 · 규정 검토 · 바이브코딩	+ 서브에이전트 · 울트라코드

반대급부 — 사용량을 더 많이 씁니다. 그래서 요금제가 중요해집니다. (2026년 11월 기준 · 가격은 claude.com/pricing)

서브에이전트 — 클로드가 여러 명으로

서브에이전트 = 클로드가 일의 일부를 떼어, **별도의 클로드에게 맡기고 결과만 받아 오는 것**

	혼자 할 때	맡길 때
속도	20개를 하나씩 차례대로	5개를 시키면 5개가 한꺼번에
내 대화창	20개 내용이 전부 쌓임	찾은 결론만 돌아옴
맥락	길어질수록 앞을 잊기 시작함	각자 자기 대화창을 따로 씀

쓰는 법 **"서브에이전트를 나눠서 동시에 해줘."** **또는 ultracode**

PART 01 · 핵심

서브에이전트는 업무에서 언제 쓰나

CASE 01

자료 여러 개를 각각 봐야 할 때

딥리서치
20개를 나눠서 동시에 요약·조사

CASE 02

데이터 정합성이 중요할 때

규모가 크고 여러 개로 나뉜 데이터를 스캔

CASE 03

관점이 여러 개 필요할 때

세 가지 관점으로 각각 검토해서 비교

서브에이전트는 따로 정하지 않으면 내 대화창과 같은 모델로 돌아갑니다. 단순한 일은 sonnet 으로 지정해 아낍니다.

울트라코드 — 서브에이전트를 만들고 지휘

울트라코드 = **서브에이전트 여러 명을 스크립트로 자동 지휘** (오케스트레이션)

	보통 모드	울트라코드
켜는 법	/effort 로 강도만 고름	/effort 에서 Ultracode 켜기 · 또는 프롬프트에 ultracode
일하는 방식	클로드 1명이 차례대로	클로드 1명이 여러 서브에이전트를 만들고 지휘
다룰 수 있는 규모	대화창이 감당하는 만큼	대화창 한도를 넘어섭니다
데이터 누락 · 환각	생길 수 있음	서로 교차검증을 시켜 줄일 수 있음 (0은 아님)
시간 · 사용량	짧음 · 보통	김 · 매우 큼

PART 01 · 핵심

울트라코드는 켤 때와 끌 때가 있습니다

켤 때

규정 · 계약 전수 대조

제출물 일괄 검토 · 틀리면 곤란한 자료

끄고 갈 때

문장 다듬기 · 짧은 질문

사용량만 크고 얻는 게 없습니다

이렇게 시킵니다

ultracode
이 규정 문서 3개를 비교해서, 서로 충돌하는 조항을 전부 찾아줘.
빠뜨린 게 없는지 다시 검증까지 해줘.

PART 01 · 핵심

왜 다룰 수 있는 규모가 커지나

보통

자료 100개를 읽으면

그 100개가 전부 대화창에 들어갑니다.

→

울트라코드

스크립트가 중간 결과를 듭니다

클로드는 결론만 봅니다.

1회차의 환각과 연결됩니다. 검증은 여전히 AI 바깥에서 합니다. 대신 검증에 걸리는 시간이 줄어듭니다.

PART 02 · 넓게

클로드를 넓힙니다

닿는 범위를 밖으로 — 스킬 · MCP · CLI · API · 힉스필드

스킬 · MCP · CLI · API — 이 순서로 봅니다

#	단어	원래 뜻	한 문장으로
1	스킬	Skill	잘 되던 긴 프롬프트를 파일로 굳혀 두고, 트리거로 부른다
2	MCP	Model Context Protocol	클로드(LLM)를 바깥 도구에 꽂는 표준 플러그
3	CLI	Command Line Interface	버튼 대신 글자로 조작할 수 있는 창 (↔ GUI, 버튼·화면으로 조작)
4	API	Application Programming Interface	사람 말고 프로그램만 드나드는 문 — AI가 아니라 '프로그램'을 위한 연결 포인트

클로드 + 스킬

내 업무 규칙·프롬프트를 붙여 둠

→

CLI · MCP

AI가 조작하면서 씀

→

API

AI가 아닌 일반 프로그램도 접근 (예: 구글 시트의 API)

PART 02 · 1

스킬 — 잘 된 프롬프트를 트리거로 소환

스킬 = 메타 프롬프팅으로 만든 길고 정교한 프롬프트를, **파일로 저장한 뒤 트리거로 소환**

스킬이 없을 때	스킬로 만들어 두면
잘 됐던 프롬프트 위치를 못 찾음	파일로 남아 항상 그 자리에
쓸 때마다 A4 한 장을 다시 붙여넣음	한마디 하면 클로드가 알아서 펴 봄

클로드가 스스로

description 한 줄을 읽고 스스로 판단

내가 직접

스킬 이름을 부른다 — /weekly-report 또는 '주간보고서 작성해 줘'

폴더 하나 + SKILL.md 하나 — 위 두 줄이 트리거

```
---
name: weekly-report
description: 주간 사업 현황 보고서를 우리 양식으로 작성할 때 사용
---
우리 회사 주간보고는 항상 이 순서로 씁니다.
1. 이번 주 완료 항목  2. 다음 주 계획  3. 리스크 · 지원 요청
- 숫자는 반드시 전주 대비 증감을 함께 씁니다.
```

코드가 아니라 말로 쓴 업무지시서(프롬프트)입니다.

PART 02 · 2

MCP — 바깥 도구에 꽂는 표준 플러그

MCP = AI와 바깥 도구를 잇는 연결 규격. **기기마다 다르던 케이블을 하나로 통일한 USB-C**와 같습니다.

	MCP가 없을 때	MCP로 붙였을 때
구글 시트	내가 열어서 복사 → 붙여넣기	클로드가 직접 열고 씁니다
사내 문서함	파일을 하나씩 올림	필요한 것을 스스로 찾아 옵니다
결과 반영	클로드 답을 다시 손으로 옮김	그 자리에서 원본이 바뀝니다

실제로 쓰는 말 1

"힉스필드 MCP 연결해줘."

실제로 쓰는 말 2

"연결된 힉스필드 MCP로 이미지 생성해서 적절하게 배치해줘."

PART 02 · 3

CLI — GUI와 무엇이 다른가

	GUI	CLI
어떻게 시키나	화면에서 버튼을 찾아 누른다	하고 싶은 일을 글로 쓴다
할 수 있는 범위	만든 사람이 버튼으로 만들어 둔 것만	글로 표현되는 것 전부
파일 100개 이름 바꾸기	100번 누른다	한 줄

GUI

엑셀 · 크롬 · 아래아한글

CLI

터미널, 클로드 코드

왜 CLI가 중요한가 — AI, 즉 LLM(언어모델)은 '글'로 소통하기 때문

CLI의 단점을 없앤 것이 클로드 코드

CLI의 단점	클로드 코드에서는
명령어를 외워야 한다	자연어로 말하면 됩니다
뭘 쓸 수 있는지 안 보인다	“이거 할 수 있어?” 라고 물으면 됩니다
한 글자만 틀려도 안 된다	대충 말해도 알아듣고 되묻습니다

GUI의 쉬움 + CLI의 범위.

API — 프로그램만 드나드는 문

API = 외부 서비스가 **프로그램을 위해 따로 열어 둔 문**.

	정문 · 화면	후문 · API
누가 쓰나	사람	프로그램
어떻게	눈으로 보고 마우스로 누른다	정해진 형식으로 주고받는다
필요한 것	아이디 · 비밀번호	열쇠 (API 키)

MCP와 비슷하지만, 'AI'와 상관없이 모든 '프로그램'에 열려 있다는 점이 다릅니다.

PART 02 · 4

오늘 03 파트에서 이렇게 이어집니다

구글 시트 "웹에 게시"	읽기만 됩니다. 주소를 아는 사람은 다 봅니다
구글 시트의 Sheets API	**읽기 + 쓰기. 열쇠가 필요합니다** 그 열쇠를 발급하는 곳이 GCP 입니다

열쇠를 화면 코드에 넣으면 브라우저 개발자도구를 여는 누구나 가져갑니다.

이미지·영상 생성 — 클로드가 못 하는 일

클로드가 하는 것

글 · 표 · 코드 · 파일 · 웹 화면

클로드가 못 하는 것

사진 · 그림 · 영상

클로드는 간단한 도형은 그리지만, 사진 같은 이미지는 직접 만들어 내지 못합니다.

그래서 막히는 지점	
교육 담당	과정 안내문 배너 · 수료증 배경 · 사내 공지용 이미지
사업 관리	제안서 표지 · 발표자료 삽화 · 홍보용 짧은 영상
지금까지는	다른 AI툴로 만들거나, 혹은 직접 찾거나 그려서 삽입

PART 02 · 핵심

힉스필드 — 이미지·영상을 한 계정에서

힉스필드(Higgsfield) = **주요 AI 이미지 · 영상 생성 모델을 한 계정에서 쓰는 곳**.

이미지

말로 설명하면 나옵니다

해상도·사양은 모델마다 다릅니다.

영상

짧은 영상

짧은 영상을 만듭니다. 길이는 모델마다 다릅니다.

캐릭터

같은 인물 · 같은 톤

여러 장에 걸쳐 유지합니다.

가격과 지원 사양은 바뀌므로 공식 사이트에서 확인합니다.

PART 03 · 남긴다

만든 것을 남깁니다

파일에서 멈추지 않습니다 — 주소를 붙이고, 데이터를 밖으로 뺍니다

PART 03

바이브코딩 — 말로 만드는 소프트웨어

코드를 쓰지 않고, 만들고 싶은 것을 **말로 설명해서** 프로그램을 만드는 방식.

예전

문법 → 코드 → 오류

문법을 배우고 → 코드를 쓰고 → 오류를 잡는다

→

바이브코딩

"이런 걸 만들어줘"

AI가 만든다 → 고쳐달라고 한다
할 수 있는 사람 — 코딩을 몰라도 '업무'를 아는 사람

금방 되는 것

내부에서 쓰는 도구 · 우리 팀 몇 명이 쓰는 수준 · 빠르게 만들고 빠르게 고치기

금방은 안 되는 것

고객에게 파는 상용 서비스 · 수천 명이 동시에 쓰는 시스템 · 보안 · 결제가 얽힌 민감한 시스템

업무를 모르는 개발자보다, 업무를 아는 사람이 더 잘 만듭니다.

PART 03 · 핵심

AI 슬롭 — 만들기 쉬워지면 생기는 것

AI 슬롭 = AI로 대량으로 찍어낸, **그럴듯한데 알맹이 없는 결과물**. 업무에서 나타나는 형태를 워크슬롭(workslop)이라 부릅니다.

약 40%
최근 한 달 안에 워크슬롭을 받아 봤다

약 2시간
한 건 정리하는 데 걸린 평균 시간

42%
보낸 사람을 전보다 덜 믿게 됐다

월 $186
직원 1인당 비용 (자가 보고 기반 추정)

출처: HBR 2025.9 · BetterUp Labs × 스탠퍼드 조사, 미국 직장인 1,150명(자가 보고)

일이 사라진 게 아니라

받는 사람에게 넘어간 것.
이해와 검토가 새로운 병목이 됩니다.

진짜 손해는 시간이 아닙니다

"이 사람이 보낸 건 한 번 더 확인해야 한다"가 되는 순간, 그 사람의 모든 산출물이 느려집니다.

PRD — 만들기 전에 정하는 것

PRD(요구사항 정의서) = 만들기 전에 **「무엇을 만들 것인가」**를 적어 둔 한 장.

1 **PRD 작성** → 2 **검토 · 확정** → 3 **개발 착수** → 4 **확인 · 수정**

준비 없이

"만들어줘"

→

PRD를 먼저 쓰고 시작

방향이 고정됩니다

고치는 게 아니라 쌓입니다

SSOT	**PRD는 이 도구의 SSOT입니다** 범위를 묻고, 이어받는 사람에게 줍니다
저장	**PRD.md 파일로 저장합니다** 워드로 쓰지 마십시오. 읽을 때만 직관적인 HTML로.

웹·앱 개발 시 필수 도구

만드는 사람

Claude Code

없으면 못 만든다

설계도 보관소

GitHub

없으면 되돌릴 수 없다

매장 오픈

Vercel

없으면 나만 쓴다

장부 (+ GCP)

Google Sheets

없으면 데이터 저장이 안 된다

파일 → 주소 → 데이터. 이 순서로 한 칸씩 올립니다.

1 만든다 → 2 올린다 → 3 주소를 받는다 → 4 데이터 관리

버전관리 · 호스팅 · 배포(deploy)

파일로 두면	주소를 붙이면
파일을 메일로 보내야 함	링크만 보내면 됨
받은 사람마다 버전이 다름	고치면 전원이 동시에 최신
컴퓨터가 죽으면 같이 사라짐	GitHub에 이력까지 남음

파일로 끝난 도구는 만든 사람만 씁니다. 조직에 남지 않습니다. 주소가 붙는 순간 '내 도구'가 '우리 도구'가 됩니다.

GitHub와 Vercel은 하는 일이 다릅니다

	하는 일
GitHub	원본 보관 + 변경 이력 · 설계도 캐비닛
Vercel	주소 부여 + 자동 배포 · 매장 오픈
관계	**GitHub에 올리면 → Vercel이 자동으로 감지해서 다시 배포합니다 (연결 후 · main 기준)**

한 번만 연결해 두면, 그다음부터는 "고쳐줘" 한마디로 전 세계가 최신이 됩니다.

PART 03 · 핵심

DB — 데이터를 어디에 둘 것인가

데이터를 수집 · 저장 · 편집 · 추출 · 다운로드 · 관리하기 위해 필요한 ‘DataBase’

1단

화면 안에 잠깐만 보임

2단 · 오늘 목표

구글 시트를 읽기 전용으로 게시

시트에서 그냥 고칩니다.

3단 · 되면 성공

구글 시트 + GCP 정식 연결

화면에서 직접 입력 · 수정
시트를 공개할 수 없을 때

대부분의 업무 도구는 2단에서 끝나도 충분합니다. 구글 시트 1장이 원본(SSOT)이 됩니다. 화면은 그걸 비출 뿐입니다.

PART 03

GCP — 시트를 제대로 붙이려면

2단 「웹에 게시」의 두 가지 한계

읽기만 됩니다

주소를 아는 사람은 봅니다

둘을 풀려면 구글이 정식으로 열어 둔 문 — **Google Sheets API**를 활성화하고 연결합니다. 열쇠는 GCP 에서 발급합니다.

막히면 클로드에게 묻고 시키세요! (메뉴 이름은 바뀝니다 · 2026년 11월 기준)

할 일은 네 가지뿐입니다

1 **프로젝트 하나 만들기**
console.cloud.google.com

2 **Google Sheets API 사용 설정**
같은 콘솔 → API 라이브러리에서 검색

3 **서비스 계정 만들고 키 받기**
같은 콘솔 → 서비스 계정 → 키 탭 (JSON)

4 **그 서비스 계정 이메일에 시트를 공유**
구글 시트 → 공유

ACTIGENCE

마무리

바로 지금부터, 개발자가 됩니다

오늘 만드신 도구를 다음 주에 한 번만 실제로 열어 보십시오.

actigence.ai · cio@actigence.ai