

AI 코딩 에이전트 실전 운용 토큰 비용 절감 전략

도구를 바꾼다고 싸지지 않는다 — 컨텍스트에 무엇을 넣느냐가 비용을 정한다

180분 · 동작 원리와 비용 구조 40 / 컨텍스트·프롬프트·티어 60 / 휴식 15 / 시연 45 / 보안·비교 평가 20 · 1부
부천 · 2026. 10.

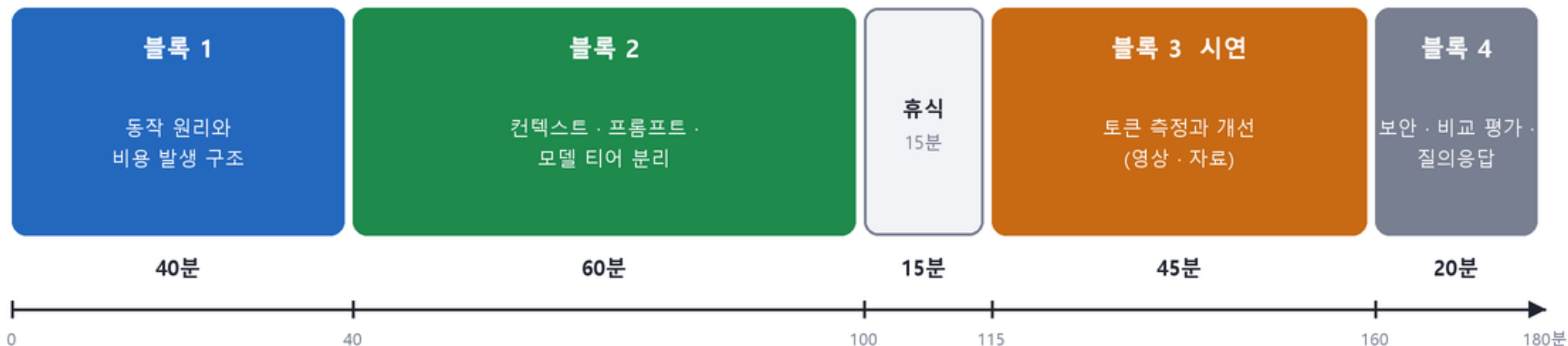


과정 구성

180분 진행 흐름 (부천 1안 커리큘럼 기준, 휴식 15분 포함)

발표자료 1부

발표자료 2부



블록	시간	이 블록에서 남길 것
1 동작 원리와 비용 발생 구조	40분	매 요청에 대화 전체가 실린다. 비용은 거기서 든다
2 컨텍스트 · 프롬프트 · 모델 티어	60분	넣을 것을 줄이고, 밖에서 끝내고, 일에 맞는 모델을 쓴다
휴식	15분	발표자료는 여기서 1부와 2부로 갈린다
3 시연 — 토큰 측정과 개선 (영상 · 자료)	45분	강사가 직접 측정한 전후 비교표 한 장. 집에서 재현하는 절차
4 보안 · 비교 평가 · 질의응답	20분	금지가 아니라 통제. 측정 없이는 최신 방식도 비용이 더 든다

BLOCK 1

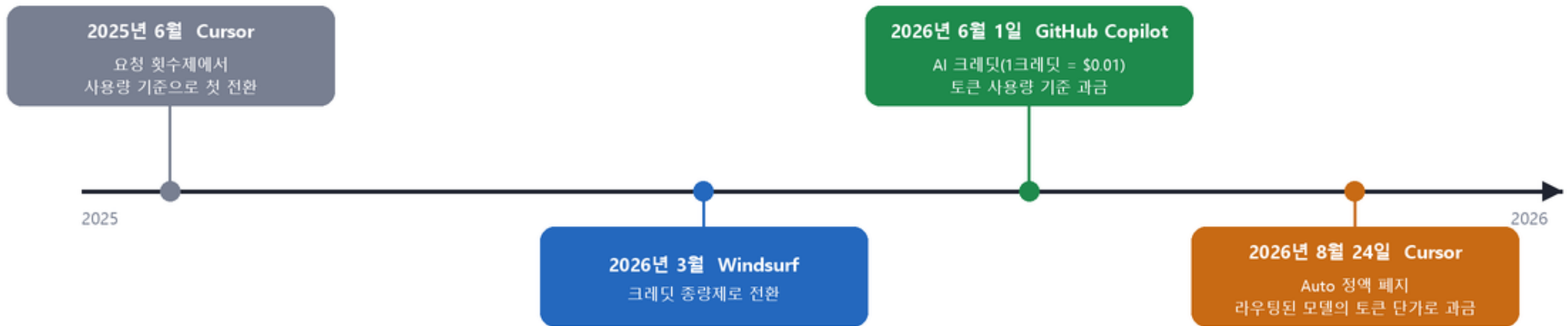
코딩 에이전트 동작 원리와 비용 발생 구조

40분 · 매 요청에 대화 전체가 실린다. 비용은 거기서 든다

1

정액제가 끝났다

정액제에서 종량제로, 코딩 에이전트 과금 전환 연표 (미국 업체). 아래 띄는 중국 업체의 반대 방향 흐름



중국 업체 (DeepSeek · Zhipu · Kimi)

챗봇은 무료, 돈은 API 종량제 → 2024년 단가 경쟁 → 2026년 코딩 월정액 구독 추가 (5시간 한도, 피크 차등)

미국은 정액에서 종량으로, 중국은 종량에서 정액으로. 가운데서 만났다

좌석료만 보고 예산을 잡으면 틀린다

월 요금 위에 토큰 요금이 얹힌다

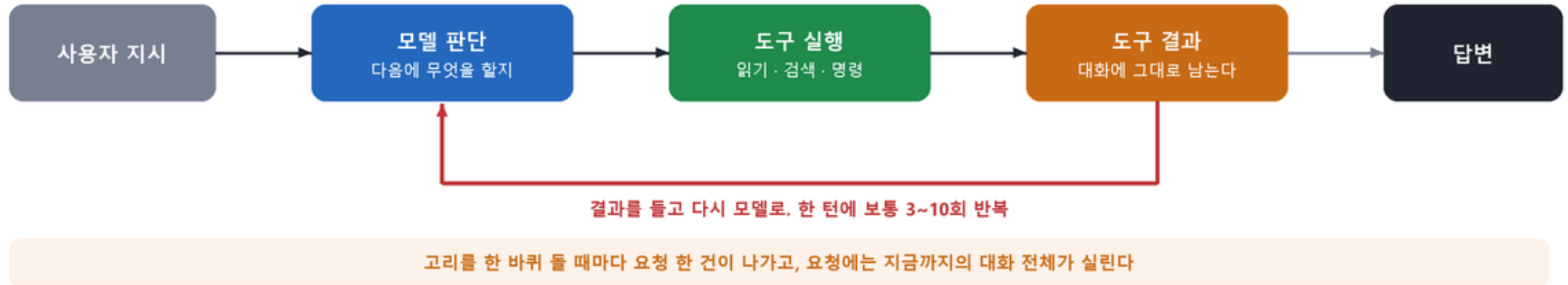
강사의 정정판

"MCP로 토큰 75% 절감" → 비교 기준이 한쪽에 불리했다

2

에이전트 한 턴의 고리

에이전트 한 턴의 고리. 도구를 부를 때마다 요청이 한 번 더 나간다



도구 = 모델이 요청 도중 부르는 기능(파일 읽기·검색·셀·서브에이전트) · 부를 때마다 요청 한 번 더 · 매 요청에 대화 전체가 다시 실린다

비용은 '내가 몇 마디 했나'가 아니라 '대화에 뭐가 쌓여 있나'에서 나온다

3

요청 한 건에 실리는 것

요청 한 건에 실리는 것. 변하지 않는 것이 앞에, 매 턴 바뀌는 것이 뒤에



도구 목록이 바뀔 때만 변함

세션 시작, /clear, /compact 때 다시 읽음

매 턴 뒤에 덧붙임

앞부분이 그대로면 캐시에서 읽는다 (정가의 5~10%, 매번)

여기만 새로 계산한다

앞쪽 층을 건드리면 그 뒤 전부를 다시 계산한다. 도구 추가, 모델 교체, 노력 수준 변경이 대표적이다

변하지 않는 것이 앞에

시스템 프롬프트 · 도구 정의 → 프로젝트 설정(CLAUDE.md)

도구만 써도 토큰이 든다

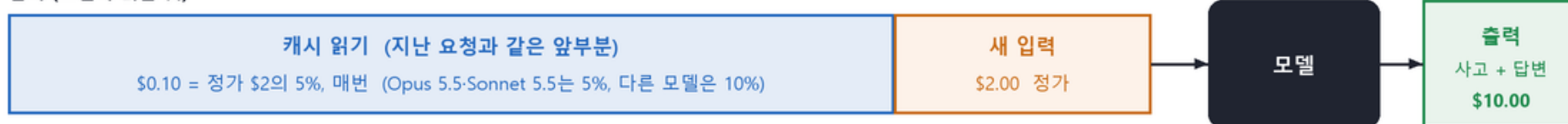
도구 사용 설명 286토큰 · 셸(Bash) 도구 정의 325토큰 · Claude Code 도구 스무 개 합계 2만 토큰 이상

4

비용이 드는 자리 — 토큰 네 종류

요청 한 건에서 비용이 드는 자리. 토큰 네 종류 (Sonnet 5.5 기준, 100만 토큰당)

입력 (모델이 읽는 것)



캐시 쓰기 (한 번만): 다음 요청을 위해 저장하는 값. 5분 캐시 정가의 1.25배(\$2.50), 1시간 캐시 2배(\$4.00). 두 번째 요청부터 이득

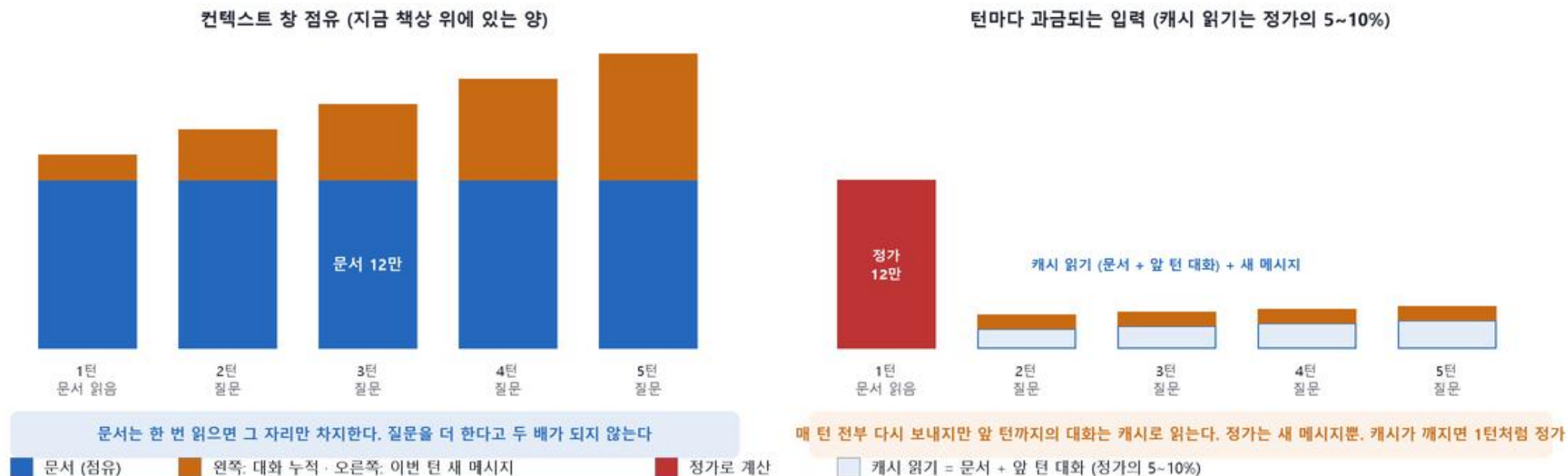
긴 세션에서 입력의 대부분은 캐시 읽기다. 캐시가 깨지면 그 전부가 정가로 돌아간다. 사고 토큰은 출력 단가로 붙는다

요청	이번에 실리는 것 (앞층 30k 세션)	캐시 읽기 (5%)	캐시 쓰기 (1.25배)	그 행의 입력
1	앞층 30k + 지시 1k	0	31k	31k
2	위 31k + 도구 결과 5k	31k	5k	36k
3	위 36k + 도구 결과 20k	36k	20k	56k
4	위 56k + 답 2k + 새 지시 1k	56k	3k	59k
합계	네 행의 입력을 더한 것 (매 요청 대화 전체가 실린다)	$123k \times 5\% = 6.2k$	$59k \times 1.25 = 73.8k$	182k
결과	182k 정가 → 캐시 있으면 약 80k 어치 (56% 절감)	Sonnet 5.5 단가로 \$0.36 → \$0.16 ($182k \times \$2/M \rightarrow 59k \times \$2.5/M + 123k \times \$0.1/M$)		

쓰기는 '처음 나타난 조각'만, 읽기는 누적 전체를 정가의 5~10%(Opus·Sonnet 5.5는 5%, Fable 5.1은 2.5%)

컨텍스트 창 점유와 누적 과금은 다르다

컨텍스트 창 점유와 누적 과금은 다르다. 문서는 한 번만 차지하고, 과금은 매 턴 불되 캐시로 할인된다



문서는 한 번만 자리를 차지한다 · 과금은 매 턴 불지만 앞부분은 캐시라 정가의 5~10% · 캐시가 깨지면 1턴처럼 정가

\$

가격표 (2026. 10. 10. 공식 가격표 기준)

모델	입력	캐시 읽기	5분 캐시 쓰기	출력
Haiku 5.5 (10만 토큰 이하 / 초과)	\$0.10 / \$0.50	\$0.01 / \$0.05	\$0.125 / \$0.625	\$0.50 / \$2.50
Haiku 4.5	\$1	\$0.10	\$1.25	\$5
Sonnet 5.5	\$2	\$0.10	\$2.50	\$10
Sonnet 5	\$2	\$0.20	\$2.50	\$10
Opus 5.5	\$4	\$0.20	\$5	\$20
Opus 5	\$5	\$0.50	\$6.25	\$25
Fable 5.1	\$10	\$0.25	\$12.50	\$50

100만 토큰당, 미국 달러 · 배치 API는 입력·출력 모두 50% 할인 · Haiku 5.5(10/7 출시)는 요청의 프롬프트가 10만 토큰을 넘으면 그 요청 전체가 5배

같은 모델 안: 캐시 읽기는 정가의 10% (Opus 5.5 · Sonnet 5.5는 5%, Fable 5.1은 2.5%)

모델 사이: Sonnet 5.5는 Opus 5.5의 절반, Haiku 5.5는 40분의 1

7

비용이 새는 일곱 군데

비용이 새는 일곱 군데. 대부분 "긴 세션"과 "기본값"에서 나온다

정리 안 한 긴 세션

한 줄 질문도 하루치 대화를
다시 듣는다

최상위 모델 기본값

탐색과 단순 수정까지
가장 비싼 단가로

캐시 미스

휴식 후 첫 메시지, 모델 교체,
도구 추가 때 전체 재계산

큰 도구 출력

테스트 로그 1만 줄이
그대로 컨텍스트에

과하게 붙인 MCP

도구 정의가 앞층에 쌓이고
바뀌면 캐시가 깨진다

서브에이전트 · 에이전트 팀

각자 컨텍스트를 갖는다.
팀은 약 7배

예약 작업 · 백그라운드

세션이 돌아도 주기마다
전체 컨텍스트를 보낸다

내 습관에 걸리는 것이 몇 개인가 — 세 개 넘으면 오늘 시연의 절감폭이 그대로 내 것이다



자사 예산 가늠하기

항목	공식 수치
개발자 1인, 쓴 날 하루 평균	약 \$13
개발자 1인, 한 달	\$150~250
하루 \$30을 넘기는 사람	열 명 중 한 명

구독제는 달러 대신 한도로 읽는다

5시간·주 단위 한도. 비싼 모델일수록 빨리 깎인다
크레딧으로 넘어가면 캐시 수명 1시간 → 5분

블록 1 정리

에이전트는 한 턴에 요청을 여러 번 보내고, 매 요청에 대화 전체가 실린다

입력의 대부분은 캐시 읽기다. 캐시가 깨지면 정가로 돌아간다

비용은 긴 세션과 기본값에서 샌다

BLOCK 2

컨텍스트 관리 · 프롬프트 설계 · 모델 티어 분리

60분 · 넣을 것을 줄이는 것이 먼저, 모델을 바꾸는 것은 마지막

1

책상은 그대로, 위만 치운다

책상 위 양	캐시 읽기, 요청 1번	요청 100번	캐시가 한 번 깨질 때
33만 토큰 (강사 기준선 세션의 요청당 평균)	약 \$0.07	약 \$6.6	약 \$2.6
3.3만 토큰 (새 세션)	약 \$0.007	약 \$0.66	약 \$0.26

강사 세션의 양을 Opus 5.5 단가로 환산 · 캐시 읽기 100만 토큰당 \$0.20 · 1시간 캐시 쓰기는 정가 입력의 두 배인 \$8

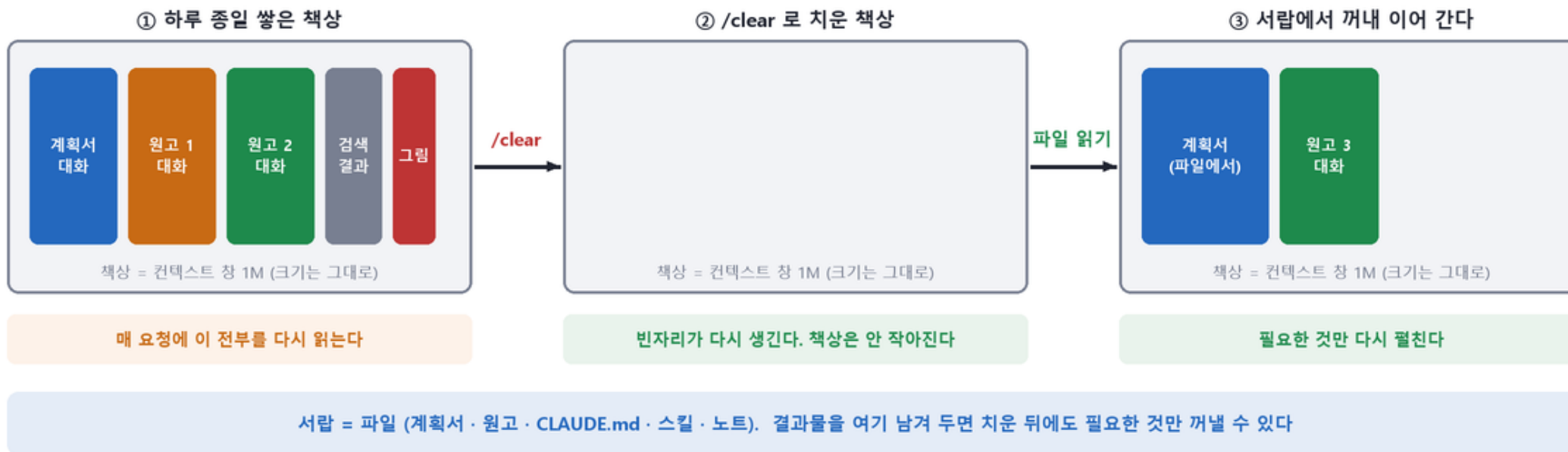
상황	할 일	이유
작업 단위가 바뀐다	/clear 후 새로 시작	앞 작업 자료가 필요 없다
같은 작업이 길어진다	/compact 에 남길 것을 적어서	무엇을 남길지 내가 고른다
엉뚱한 방향으로 갔다	/rewind	이미 캐시된 지점으로 돌아가서 싸다
어제 하던 일을 잇는다	파일을 다시 읽게 하거나 /resume	서랍에서 필요한 것만 꺼낸다
큰 코드베이스 전체를 한 번에 봐야 한다	1M을 그대로 쓴다	큰 책상이 필요한 순간이다

/clear 는 책상을 줄이지 않는다 · 이어 가기는 서랍(파일)이 한다

1

책상은 그대로, 위만 치운다 — 그림으로

책상은 그대로, 위만 치운다. 이어 가기는 서랍(파일)이 한다



/clear 는 책상 위를 치우는 것 · 책상(1M)은 안 작아진다 · 이어 가기는 서랍(파일)이 한다

1

비우기 전후 실측 — 54만이 5만으로

항목	비우기 전 (10-11 20:22)	비운 뒤 (20:25)
/context 총합	542.3k / 1M (54%)	50.9k / 1M (5%)
앞층 — 시스템 프롬프트 4.4k + 도구 정의 23.2k + MCP 안내 1k + 메모리 2.1k + 스킬 6.6k	37.3k	37.3k (그대로)
대화층 (메시지)	505k	13.5k
요청 1번의 캐시 읽기 — Fable 5.1 (\$0.25/M)	542k → 약 \$0.14	51k → 약 \$0.013
같은 양을 Opus 5.5로 (\$0.20/M)	약 \$0.11	약 \$0.010
요청 100번 — Opus 5.5	약 \$10.8	약 \$1.0
캐시가 한 번 깨질 때 — Opus 5.5, 1시간 쓰기 \$8/M	약 \$4.3	약 \$0.4

강사 세션 실측 · 문서 작업 폴더(원고·docx·pptx·그림, git 아님, 프로젝트 CLAUDE.md 없음) · 10-10 밤 압축 뒤 6.6만에서 하루 190회 요청으로 54만까지

책상이 50만을 넘은 뒤 23회 — 요청당 \$0.22, 그중 캐시 읽기 \$0.13, 출력은 평균 1.2k → 쓴 것보다 읽은 것이 비싼 세션
Read 결과 누적 3.7M 토큰(창의 366%) — 원고 md 통째 읽기 + 렌더 검증 PNG 119장. 코드 저장소면 테스트 로그와 긴 소스가 그 자리
비운 뒤 첫 요청 \$0.38 — 새 대화 16.5k 캐시 쓰기 · 앞층 37.3k 중 34k(캐시 경계)는 앞 세션 캐시가 그대로 맞음 · 둘째부터 \$0.11~0.14

앞층 37.3k 는 그대로 · 차이는 전부 대화층 · 캐시 읽기가 54만에서 5만으로, 요청당 비용은 \$0.38 에서 \$0.11~0.14 로

2

캐시 접두부 보존

네 번의 턴. 앞부분이 같으면 캐시에서 읽고, 앞층이 바뀌면 전부 다시 계산한다

1턴

전부 새로 계산

2턴

캐시 읽기

새 교환

3턴

캐시 읽기

새 교환

4턴 도구 정의 변경

앞층이 바뀌어 전부 다시 계산 (정가 + 캐시 쓰기)

캐시는 5분(API · 크레딧) 또는 1시간(구독) 쉬면 사라진다. 휴식 후 첫 메시지는 4턴과 같은 값을 치른다

앞부분이 같으면 캐시에서 읽고, 앞층이 바뀌면 전부 다시 계산한다

캐시 수명 — 구독 본 대화 1시간 · API 키·사용 크레딧·서브에이전트 5분

확인 — /usage 의 "Prompt cache" 줄 (히트율, 미스 횟수, 마지막 미스 원인)

3

캐시를 깨는 행동

캐시를 깨는 행동	왜
모델 교체 (/model)	모델마다 캐시가 따로 있다. 내용이 같아도 전부 다시 읽는다
노력 수준 변경 (/effort)	대부분 모델에서 수준마다 캐시가 따로. Opus 5.5·Sonnet 5.5·Haiku 5.5·Fable 5.1은 예외(API·구독일 때)
패스트 모드 켜기	요청 헤더가 캐시 키에 들어간다. 세션 초반에 켜는 게 싸다
MCP 서버 연결·해제	도구 정의가 앞층이다. 지연 로딩이 켜져 있으면 괜찮고, 아니면 전부 다시
도구 통째로 거부 규칙 추가	위와 같은 이유. 지연 로딩이면 괜찮다
플러그인 설치·제거	도구·스킬 목록이 바뀐다. 앞층이 바뀌니 전부 다시
압축	대화층이 요약본으로 바뀐다. 의도된 재계산
이미지가 많이 쌓임	한도에 닿으면 오래된 이미지 한 묶음을 뺀다. 그 메시지가 바뀌므로 거기서부터 뒤를 다시 계산
Claude Code 업그레이드	시스템 프롬프트가 바뀐다. 새 세션 첫 턴에 한 번

지키는 행동 — 파일 편집 · CLAUDE.md 편집 · 권한 모드 변경 · 스킬 호출 · 되감기 · 서브에이전트 생성

4

실행 위임 — 컨텍스트 밖에서 끝내기

컨텍스트 밖에서 일을 끝내는 네 가지 수단. 모델에게 들어가는 것만 다르다



모델이 읽을 것과 PC가 돌릴 것을 나눈다 · 스크립트 출력은 요약해서 돌려준다

5

실행 위임 네 수단

수단	모델과 만나는 방식	평소 컨텍스트 비용	언제 쓰나
스킬	지시문을 읽고 스크립트는 셸로 실행만 한다	이름과 설명 한 줄. 본문은 호출할 때만	반복 절차, 문서 생성, 검증 스크립트
훅	이벤트가 나면 자동으로 명령이 돈다. 모델은 결과만 본다	실행 때 토큰 0 (만들 때는 든다)	테스트 출력 거르기, 커밋 전 검사, 정책 차단
MCP	서버가 도구 스키마를 선언하고 모델이 직접 호출한다	도구 이름 목록. 정의는 쓸 때 들어온다	사내 시스템, 브라우저, 인증이 필요한 외부 서비스
코드 지도	미리 만든 파일을 프롬프트에 끼워 넣는다	관련 노드만	큰 저장소 탐색, 영향 범위 파악

MCP는 토큰을 아끼는 도구가 아니라 셸로 닿지 않던 곳에 손을 뻗게 하는 도구

Graft(코드 지도) 제작사 벤치마크: 토큰 42% 절감, SWE-bench 23% — 숫자보다 원리. 파싱은 PC가, 모델은 결과 파일만

5

흑이란 — 모델이 아니라 프로그램에 거는 규칙

흑은 모델이 아니라 Claude Code 라는 프로그램에 걸어 두는 규칙이다. 테스트 로그 5,932줄은 모델이 한 번도 보지 않는다

모델 컨텍스트에 들어가는 것 (토큰이 든다)

① 모델

"npm test 를 실행해"
(Bash 도구 호출)

모델은 흑이 있는지 모른다. 자기가 시킨 npm test 의 결과가 4줄로 돌아온 것만 본다

흑이 없으면 5,932줄이 그대로 대화에 붙어 매 요청마다 다시 읽힌다

⑥ 모델이 받는 결과

실패 줄과 요약 4줄
(수백 토큰)

호출

PC 에서 도는 것 (토큰 0) — Claude Code 가 흑을 실행하고, 흑이 명령을 바꾼다

② Claude Code

도구를 돌리기 직전에
settings.json 의 PreToolUse
흑을 먼저 실행한다

③ filter-test-output.js

명령을 JSON 으로 받아
"npm test" 면 뒤에
| only-failures.js 를 붙인다

바뀐 명령

④ 테스트 실행

npm test 2>&1
출력 5,932줄

⑤ only-failures.js

fail · error · expected
· 요약 줄만 남긴다
최대 100줄

4줄만

켜는 법 : 샘플 저장소의 .claude/settings.with-hook.json 이름을 settings.json 으로 바꾼다 · 측정 : 5,932줄 → 4줄 (2026-10-03)

모델은 흑이 있는지 모른다 · 5,932줄은 PC 에서 걸러져 모델에게는 4줄만 간다 · 켜는 법과 파일 세 개는 2부 시연 항목 7에서

스킬은 모델이 골라 쓴다 (빠뜨릴 수 있다) · 흑은 규칙대로 매번 돈다 (막을 수도 있다) → 빠뜨려도 되면 스킬, 빠지면 안 되면 흑

흑은 측정 뒤의 처방 — 같은 긴 출력이 세션마다 반복될 때만 · 실행 때 토큰 0 (만들 때는 든다)

6

프롬프트 설계 — 재작업이 가장 큰 숨은 비용

과제 A	요청 수	출력 토큰	환산 비용	걸린 시간	원하던 답
막연한 문장 (기준 설정)	10	2,947	\$0.29	39초	아니다. 로그인 흐름 전체를 길게 설명했다
구체적 문장 (기준 설정 그대로)	2	229	\$0.19	15초	맞다. 함수 이름과 파일 경로
구체적 문장 + 블록 3 개선 설정	3	657	\$0.17	29초	맞다. 함수 이름과 파일 경로

2026-10-04 사전 측정 · 헤드리스 실행 · Opus 5.5 · 둘째 줄이 문장만 바꾼 효과, 셋째 줄은 개선 설정까지 넣은 것

① 먼저 계획

플랜 모드부터 시작
(Shift+Tab)

② 범위 지정

파일과 함수를 지정한다

③ 검증 기준

테스트 이름 · 기대 출력을
같이 준다

④ 조기 중단

틀어지면 Esc, 그리고
/rewind

하지 말 것 — 습관적인 "자세히 설명해 줘" · 긴 로그 통째로 붙이기 · 한 세션에서 무관한 일 번갈아 하기

7

모델 티어 분리

일의 종류로 모델을 나눈다. 설계는 위에서, 실행은 아래에서



계획은 위, 실행은 아래

방향은 Opus 플랜 모드, 실행은 Sonnet

저가 모델 90% 이상은 금물

절반에서 시작해 측정하며 늘린다

모델을 낮추기 전에 노력 수준부터

/effort · 서브에이전트는 model: haiku



체크리스트 — 돌아가서 세 가지 이상 고른다

번호	항목	어디서 확인
1	작업 바뀔 때 세션 비우기	/usage 세션 비용
2	압축할 때 남길 것 지시하기	/compact 지시문
3	모델·노력 수준은 세션 시작 때 정하기	/usage 캐시 미스 원인
4	CLAUDE.md 200줄 아래, 절차는 스킬로	/context
5	안 쓰는 MCP 서버 끄기, CLI 우선	/mcp, /context
6	탐색은 서브에이전트, 모델은 haiku	서브에이전트 설정
7	긴 출력은 흑으로 거르기	흑 설정
8	복잡한 일은 플랜 모드부터	Shift+Tab
9	검증 기준을 지시에 넣기	재요청 횟수
10	단순 작업은 노력 수준 낮추기, 구현은 Sonnet 5.5로	/effort, /model
11	코드 지도 프로그램 써 보기 (선택 · 다른 항목과 섞지 말고 따로)	도구 호출 횟수

휴식 15분

2부는 시연입니다 — 같은 과제를 두 번 돌려 전후를 봅니다

쉬는 동안 할 것

- 샘플 저장소 압축본 받기 (집에서 30분 재현용)
- 체크리스트에서 돌아가서 고를 세 가지에 표시해 두기