

AI 코딩 에이전트 실전 운용 토큰 비용 절감 전략 — 2부

1부에서 배운 체크리스트를 숫자로 확인한다 — 시연, 보안, 비교 평가

시연 — 토큰 측정과 개선 45 / 보안·비교 평가·질의응답 20
부천 · 2026. 10.

과정 구성 — 지금 여기부터 블록 3

180분 진행 흐름 (부천 1안 커리큘럼 기준, 휴식 15분 포함)



블록	시간	이 블록에서 남길 것
1 동작 원리와 비용 발생 구조	40분	1부에서 마쳤다. 매 요청에 대화 전체가 실린다. 비용은 거기서 든다
2 컨텍스트 · 프롬프트 · 모델 티어	60분	1부에서 마쳤다. 체크리스트 11개를 나눠 줬다
휴식	15분	여기서 1부와 2부로 갈렸다
3 시연 — 토큰 측정과 개선 (영상 · 자료)	45분	지금부터. 체크리스트 4·5·6·7·9를 샘플 저장소에 적용해 전후 숫자를 본다
4 보안 · 비교 평가 · 질의응답	20분	금지가 아니라 통제. 측정 없이는 최신 방식도 비용이 더 든다

BLOCK 3

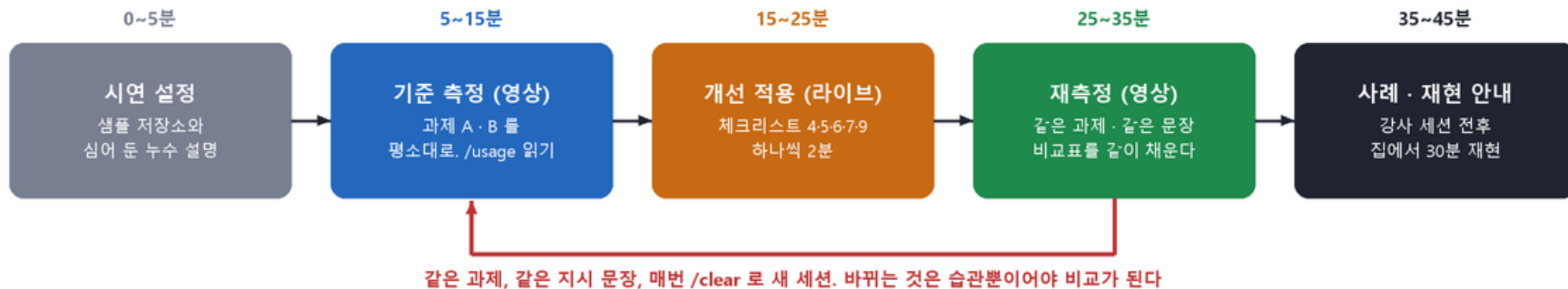
시연 — 토큰 측정과 개선 (영상 · 자료)

45분 · 강사가 같은 과제를 두 번 돌려 전후를 보인다. 집에서 재현하는 절차를 가져간다

1

시연 45분의 흐름

시연 45분의 흐름. 강사가 같은 과제를 두 번 돌리고, 사이에 습관 다섯 가지를 바꿔 보인다



수강생은 워크시트에 강사 화면의 숫자를 따라 적는다 : 요청 수 · 캐시 적중률 · 입력/출력/캐시 토큰 · 환산 비용 (모두 /usage 한 화면)

규칙 1

같은 과제, 같은 지시 문장 — 과제 카드를 복사해 붙인다

규칙 2

과제마다 /clear 로 새 세션

규칙 3

과제가 끝나면 바로 /usage 를 열어 숫자를 적는다

2

샘플 저장소 token-diet-sample — 심어 둔 누수

심어 둔 누수	저장소 안의 모습	체크리스트 번호 (1부 23장)
두꺼운 설정 파일	CLAUDE.md 300줄. 배포 절차, 코드 리뷰 규칙, 회의록까지 들어 있다	4
과하게 붙인 MCP	.mcp.json에 과제와 상관없는 서버 세 개	5
큰 도구 출력	테스트 18개가 다 통과해도 디버그 로그가 6천 줄 가까이 쏟아진다	7
탐색용 모델 미지정	서브에이전트 정의가 없다. 탐색도 본 모델이 한다	6
막연한 요청 습관	과제 카드의 "기준 문장"이 일부러 막연하다	8, 9

과제	기준 문장 (막연함)	개선 문장 (범위 · 검증 기준 포함)
A 탐색	"로그인이 실패하면 어떤 일이 일어나는지 알려줘"	"src/auth 아래에서 로그인 실패 시 응답 메시지가 만들어지는 위치를 찾아 함수 이름과 파일 경로만 알려줘"
B 수정	"회원 가입에 입력 검증 좀 추가해줘"	"src/users/register.ts의 register 함수에 이메일 형식과 빈 비밀번호 검증을 추가해. tests/users 의 기존 테스트가 모두 통과해야 해"



/usage 읽는 법 — 영상 끝 숫자를 워크시트 어느 칸에

Session Claude Code 에서 /usage 를 친 화면 (예시)

```
Total cost:          $0.58 ⑧
Total duration (API): 1m 09s
Total code changes:   0 lines added, 0 lines removed
Usage by model:
  claude-opus-5-5: 6.9k input, 1.7k output, ④ ⑤
                  0 cache read, 69.4k cache write ⑥ ⑦
Prompt cache (main): 2 requests · 0% of input tokens ① ②
  from cache · 1 miss (likely cause: first request) ③
  · warm (1h TTL)
```

워크시트 기록 항목 (같은 번호 칸에 적는다)

① 요청 수	requests
② 캐시 적중률 %	tokens from cache
③ 캐시 미스 횟수	miss
④ 입력 토큰	input
⑤ 출력 토큰	output
⑥ 캐시 읽기 토큰	cache read
⑦ 캐시 쓰기 토큰	cache write
⑧ 세션 환산 비용 \$	Total cost

구독 로그인이면 이 블록 아래에 한도 막대가 더 보인다 · /clear 하면 지워지니 과제가 끝나자마자 적는다

왼쪽은 Claude Code 의 /usage 화면 (형식만 따른 예시, 숫자는 의미 없음) · 오른쪽은 워크시트 행 · 같은 번호끼리 옮긴다



영상 1 · 기준 A — 막연한 질문, 긴 설명 한 화면

입력 : 로그인 실패하면 어떤 일이 일어나는지 알려줘 (기준 문장 · 범위도 검증 기준도 없다)

```

Claude Code v2.1.296
Opus 5.5 with medium effort · Claude Max
D:\work\try\강의\AI 코딩 에이전트 실전 운용 토큰 비용 절감 전략\scripts\demo\_wor...

| [*] [*] [*] · Share Claude Code and earn $10 in usage credits · /passes

> /model
  L Set model to Opus 5.5 and saved as your default for new sessions

> 로그인 실패하면 어떤 일이 일어나는지 알려줘

Searching for catch|HttpError|tooMany|unauthorized
  L "catch|tooMany|unauthorized|LOGIN_"

* Brewing... (16s · ↓ 853 tokens · thinking with medium effort)
  L Tip: See an artifact you'd like to build on? Duplicate, in its title menu, gives you your own editable copy.

> |

▶▶ accept edits on (shift+tab to cycle) · esc to interrupt · ← for agents
  
```

워크시트 칸	기준 A
① 요청 수	5
② 캐시 적중률 %	92%
③ 캐시 미스 횟수	0
④ 입력 토큰	516
⑤ 출력 토큰	2.2k
⑥ 캐시 읽기 토큰	276.9k
⑦ 캐시 쓰기 토큰	20.0k
⑧ 세션 환산 비용 \$	\$0.26
결과가 맞았나	× (설명만 길다)
적용한 개선 항목	없음 (기준)

요청 5 · 출력 2.2k · 캐시 읽기 277k · \$0.26 · 1분 8초 · 답: 로그인 흐름 전체를 설명 (원하던 답 아님)

슬라이드쇼에서 화면을 클릭하면 재생 · 안 돌면 폴더에서 열기: video/영상1-기준A_261011_1744.mp4



영상 2 · 기준 B — 파일 5개 수정, 테스트 두 번

입력 : 회원 가입에 입력 검증 좀 추가해줘 (기준 문장 · 어느 파일, 어떤 검증, 무엇으로 확인할지가 없다)

```

강의노트 2 x + v 영상 2 · 기준B 기준 측정 · 과제 B 수정 x
me: '등급' });
47 assert.equal(u.grade, 'basic');

• Update(docs\api.md)
  L Added 1 line, removed 1 line
  2
  3 | 메서드 | 경로 | 설명 |
  4 |-----|-----|-----|
  5 -| POST | /users | 회원 가입. 본문 { email, password, name } |
  5 +| POST | /users | 회원 가입. 본문 { email, password, name }. 이메일 형식 (254자 이
  +하), 비밀번호 8~128자, 이름 1~50자. 위반 시 400 (INVALID_INPUT, INVALID_EMAIL, IN
  +VALID_PASSWORD, INVALID_NAME) |
  6 | GET | /users/me | 내 프로필. Authorization: Bearer 토큰 |
  7 | POST | /auth/login | 로그인. 본문 { email, password } |
  8 | POST | /auth/logout | 로그아웃 |

Running 1 shell command · 9s...
L $ node --test tests/

• Billowing... (36s · ↓ 4.3k tokens)
  L Tip: Use /btw to ask a quick side question without interrupting Claude's current work
  o medium · /effort

> |

▶▶ accept edits on (shift+tab to cycle) · esc to interrupt · ← for agents

```

워크시트 칸	기준 B
① 요청 수	9
② 캐시 적중률 %	95%
③ 캐시 미스 횟수	0
④ 입력 토큰	524
⑤ 출력 토큰	5.0k
⑥ 캐시 읽기 토큰	508.4k
⑦ 캐시 쓰기 토큰	25.2k
⑧ 세션 환산 비용 \$	\$0.41
결과가 맞았나	○ (테스트 통과)
적용한 개선 항목	없음 (기준)

요청 9 · 출력 5.0k · 캐시 읽기 508k · \$0.41 · 1분 39초 · 답: 테스트 통과, 55줄 추가 (config · register · router · 테스트 · 문서)

슬라이드쇼에서 화면을 클릭하면 재생 · 안 돌면 폴더에서 열기: video/영상2-기준B_261011_1748.mp4



영상 3 · /context — 질문 하나 하기 전에 이미 38k

입력 : /context (질문을 하나도 하기 전, 설정 파일과 도구 목록이 차지한 자리를 본다)

```

System prompt: 2.6k tokens (0.3%)
System tools: 24.5k tokens (2.5%)
MCP server instructions: 749 tokens (0.1%)
Memory files: 5k tokens (0.5%)
Skills: 5.1k tokens (0.5%)
Messages: 10 tokens (0.0%)
Free space: 929.1k (92.9%)
Autocompact buffer: 33k tokens (3.3%)

Auto-compact window: 1m tokens

MCP tools · /mcp (loaded on-demand)
└ 26 tools · 0 tokens

Memory files · /memory
└ 1 file · 5k tokens

Skills · /skills
└ 29 skills · 5.1k tokens

/context all to expand

medium · /effort

>

>> accept edits on (shift+tab to cycle) · ← for agents
  
```

총 37.9k = Claude Code 자체 32k (시스템 2.6k · 도구 24.5k · 스킬 5.1k) + CLAUDE.md 5k + MCP 안내문 0.7k · MCP 도구 26개는 0

슬라이드쇼에서 화면을 클릭하면 재생 · 안 돌면 폴더에서 열기: video/영상3-context_261011_1643.mp4

7번

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 법 : settings.with-hook.json → settings.json

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)



영상 4 · 개선 A — 서버에이전트 한 번, 두 줄 답

입력 : explorer 서버에이전트로 먼저 찾아서, src/auth 아래에서 로그인 실패 시 응답 메시지가 만들어지는 위치를 찾아 함수 이름과 파일 경로만 알려줘

```

> explorer 서버에이전트로 먼저 찾아서, src/auth 아래에서 로그인 실패 시 응답 메시지가 만들어지는 위치를 찾아 함수 이름과 파일 경로만 알려줘

• explorer(로그인 실패 메시지 위치 찾기)
  ↳ Backgrounded agent (↓ to manage · ctrl+o to expand)

• explorer 서버에이전트에게 src/auth 아래를 찾아보라고 맡겼습니다. 아직 찾는 중이고, 끝나면 함수 이름과 파일 경로만 알려드리겠습니다.

* Waiting for 1 background agent to finish

• Agent "로그인 실패 메시지 위치 찾기" finished · 5s

• explorer 서버에이전트가 찾은 결과입니다.

  - 메시지를 만드는 곳: buildLoginFailure 함수, src/auth/messages.ts:5
  - 그 메시지를 HttpError로 던지는 곳: login 함수, src/auth/login.ts:12, src/auth/login.ts:20

* Crunched for 13s · done 오후 5:51

> |

▶▶ accept edits on (shift+tab to cycle) · /tasks to see subagents · ← for agents
  
```

워크시트 칸	개선 A
① 요청 수	3
② 캐시 적중률 %	92%
③ 캐시 미스 횟수	0
④ 입력 토큰	1.0k
⑤ 출력 토큰	544
⑥ 캐시 읽기 토큰	201.3k
⑦ 캐시 쓰기 토큰	10.3k
⑧ 세션 환산 비용 \$	\$0.14
결과가 맞았나	○ (함수·경로)
적용한 개선 항목	4·5·6·7·9

요청 3 · 출력 544 · 캐시 읽기 201k · \$0.14 · 45초 · 답: buildLoginFailure, src/auth/messages.ts:5 (맞음)

슬라이드쇼에서 화면을 클릭하면 재생 · 안 돌면 폴더에서 열기: [video/영상4-개선A_261011_1751.mp4](https://video.notion.so/영상4-개선A_261011_1751.mp4)



영상 5 · 개선 B — 파일 하나, 테스트 한 번

입력 : src/users/register.ts의 register 함수에 이메일 형식과 빈 비밀번호 검증을 추가해. tests/users 의 기존 테스트가 모두 통과해야 해

```

다. ');
20   const user: User = {
21     id: newId(), email, name: String(input.name ?? '').trim(),
    • Update(src\users\register.ts)
      L Added 1 line, removed 1 line
19   if (findByEmail(email)) throw conflict('EMAIL_TAKEN', '이미 가입된 이메일입니
      다. ');
20   const user: User = {
21     id: newId(), email, name: String(input.name ?? '').trim(),
22 -   passwordHash: hashPassword(String(input.password ?? '')), grade: 'basic', cr
      -eatedAt: now(),
22 +   passwordHash: hashPassword(password), grade: 'basic', createdAt: now(),
23   };
24   save(user);
25   log.debug('user registered', { id: user.id });

Running 1 shell command...
L $ npm test 2>&1 | Select-Object -Last 15

* Razzle-dazzling... (19s · ↓ 1.7k tokens)

> |

▶▶ accept edits on (shift+tab to cycle) · esc to interrupt · ← for agents
  
```

워크시트 칸	개선 B
① 요청 수	6
② 캐시 적중률 %	95%
③ 캐시 미스 횟수	0
④ 입력 토큰	518
⑤ 출력 토큰	2.1k
⑥ 캐시 읽기 토큰	296.0k
⑦ 캐시 쓰기 토큰	13.3k
⑧ 세션 환산 비용 \$	\$0.21
결과가 맞았나	○ (18개 통과)
적용한 개선 항목	4·5·6·7·9

요청 6 · 출력 2.1k · 캐시 읽기 296k · \$0.21 · 1분 · 답: register.ts 8줄, 18개 통과 · 흑이 돈 곳 : 테스트 결과가 모델에게 4줄로 갔다

슬라이드쇼에서 화면을 클릭하면 재생 · 안 돌면 폴더에서 열기: video/영상5-개선B_261011_1752.mp4

3

시연 영상의 숫자 — 같은 과제를 두 번, 사이에 습관만 바꿨다

실행	원하던 답	요청 수	출력 토큰	환산 비용
기준 A	아니다	5	2.2k	\$0.26
개선 A	맞다	3	544	\$0.14
기준 B	맞다	9	5.0k	\$0.41
개선 B	맞다	6	2.1k	\$0.21

요청 수 = 질문 한 번에 Claude Code 가 모델 서버에 다녀온 횟수 (파일 읽기·검색마다 한 번 더) · 환산 비용 = 쓴 토큰을 API 단가로 계산한 값 (/usage 의 Total cost)
 2026-10-11 영상 네 편의 /usage 값 · Opus 5.5 · 노력 수준 medium · 실행마다 저장소를 새로 복제
 뭣은 문장(9)이 가장 크고, 설정 4·5·6·7 은 그 위에 조금 더

환산 비용 \$0.67 → \$0.35 (48% 감소)

두 과제 합계 · 바뀐 것은 체크리스트 4·5·6·7·9 뿐

요청 수 14번 → 9번 (5+9 → 3+6) · 출력 7.2k → 2.6k

두 과제 합계 · 막연한 A는 긴 설명 한 화면, 막연한 B는 파일 5개
수정과 테스트 두 번

변화율 = 개선 ÷ 기준 - 1 (예: $0.35 \div 0.67 - 1 = -48\%$)

3 예상과 실제 — 촬영 전에 적어 둔 기대 범위와 영상 값

실행	요청 수 예상 → 실제	출력 토큰 예상 → 실제	환산 비용 예상 → 실제	예상한 답	실제 답	판정
기준 A	4~14 → 5	2,000 이상 → 2.2k	\$0.20~0.40 → \$0.26	긴 설명 (함수 이름이 아님)	로그인 흐름 전체를 설명	예상대로
기준 B	8~18 → 9	4,000 이상 → 5.0k	\$0.35~0.70 → \$0.41	파일 4개 이상 · 테스트 2번 이상	파일 5개 · 테스트 2번 · 통과	예상대로
개선 A	2~5 → 3	1,000 이하 → 544	\$0.10~0.25 → \$0.14	buildLoginFailure 와 messages.ts	buildLoginFailure, messages.ts:5	예상대로
개선 B	4~12 → 6	3,000 이하 → 2.1k	\$0.15~0.30 → \$0.21	register.ts 하나 · 테스트 통과	register.ts 하나 · 18개 통과	예상대로
두 과제 합계	절반 가까이 감소 → 14→9 (-36%)	절반 이하로 → 7.2k→2.6k (-64%)	절반 가까이 → \$0.67→\$0.35 (-48%)	개선 쪽은 모두 맞는 답	개선 A·B 모두 맞음	예상대로

기대 범위는 촬영 전에 작업지시서에 '합격 기준'으로 적어 둔 값이다 · 범위 밖이면 다시 찍기로 했고, 네 편 다 한 번에 범위 안에 들어왔다
요청 수는 네 편 다 범위 아래쪽에 왔다 · 모델 답은 매번 달라서 집에서 측정하면 범위 안 다른 자리에 떨어질 수 있다

결론 : 같은 과제에서 습관 다섯 가지만 바꾸면 비용이 절반 가까이 준다는 예상이 영상에서 그대로 나왔다

4

강사 사례 — 이 강의 자료를 만든 세션

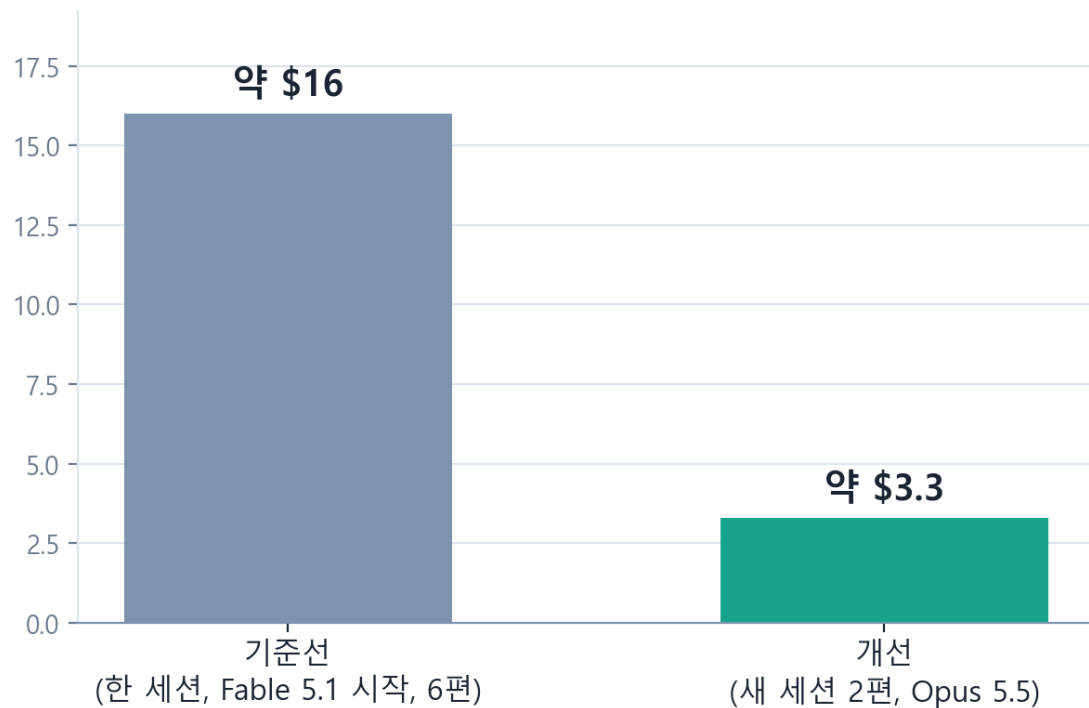
항목	기준선	개선
세션	한 세션에서 계속 작업 (10/3 22:21 시점 고정)	작업마다 새 세션
모델	Fable 5.1로 시작, 도중에 Opus 5.5	Opus 5.5
산출물	계획서, 원고 1·2·3, 비교 엑셀, 샘플 저장소 (6편)	원고 4, 발표자료 (2편)
요청 수	381 (한 편당 64)	70 (한 편당 35)
요청 한 번에 실린 평균 토큰	약 33만	약 8.2만
산출물 한 편당 환산 비용	약 \$16	약 \$3.3
비용에서 가장 큰 덩어리	캐시 쓰기 (절반 가까이)	캐시 쓰기 (절반 가까이)

같은 사람, 같은 종류의 일 — 세션을 나누고 읽을 것을 정해 주면 요청 하나의 무게가 4분의 1로

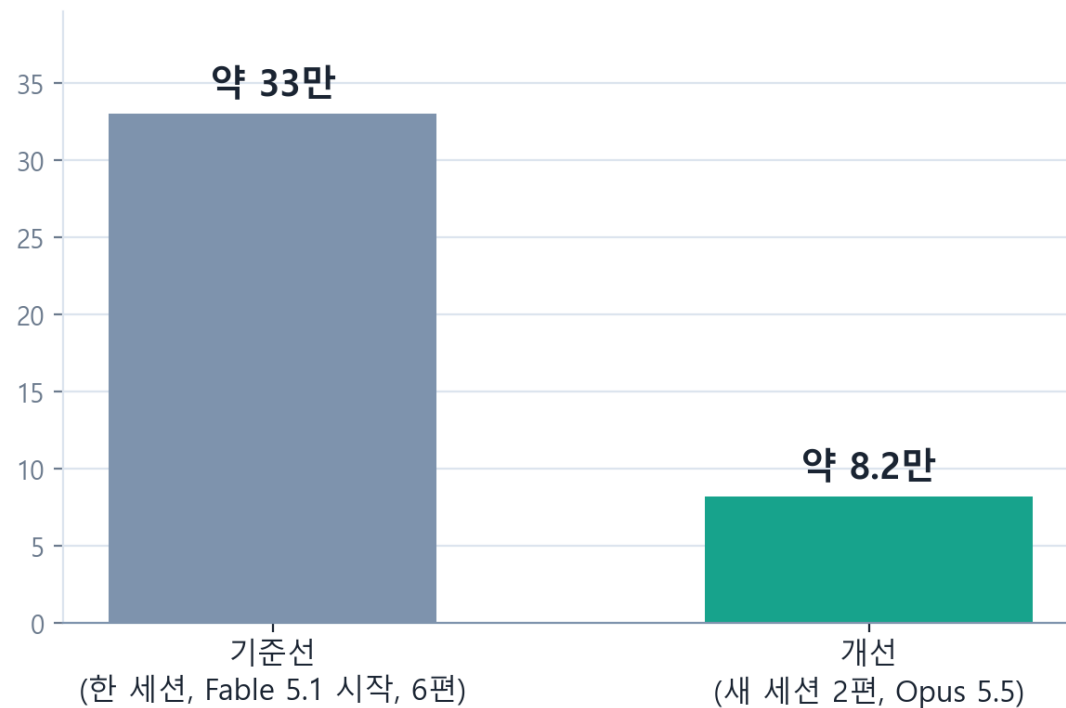
5

강사 사례 — 기준선과 개선

산출물 한 편당 환산 비용 (달러)



요청 한 번에 실린 평균 토큰 (만)



요청 수 381 → 70 (한 편당 64 → 35) · 비용에서 가장 큰 덩어리는 양쪽 모두 캐시 쓰기 · 2026-10-03 22:21 시점 고정

기준 측정 없이 고른 개선은 효과를 모른다 · 결과가 맞았는지를 같이 적는다

BLOCK 4

보안 · 비교 평가 · 질의응답

20분 · 금지가 아니라 통제. 측정 없이는 최신 방식도 비용이 더 든다

1

유출 경로 여섯 가지

경로	무슨 일이 생기나
1. 코드 전송 범위	읽은 파일과 명령 출력이 그대로 모델 쪽으로 간다. 무엇을 읽게 했는지 모르면 무엇이 나갔는지도 모른다
2. 시크릿 파일 읽힘	.env, 키 파일, 설정 파일을 에이전트가 탐색 중에 읽는다. 열지 말라고 말한 적이 없으면 연다
3. 공개 저장소의 MCP 설정 파일	토큰을 평문으로 적은 MCP 설정 파일이 커밋돼 공개 저장소에 올라간다
4. 서드파티 MCP 서버	남이 만든 서버가 받은 내용을 어디로 보내는지 확인하지 않고 붙인다
5. 프롬프트 인젝션	이슈, 웹 페이지, 외부 문서처럼 믿을 수 없는 내용을 읽고 그 안의 지시를 따른다
6. 프로그램 자체의 유출	에이전트 프로그램이 배포 과정에서 새는 경우. 공급망 문제다

3번 AI 보조 커밋 시크릿 유출률 3.2%

사람만 쓴 커밋 1.5%의 두 배 (GitGuardian 2026)

6번 2026. 3. 31. 소스맵 배포 사건

npm 패키지에 소스맵이 실려 소스 51만 줄이 드러났다

2

대책 계층

계층	하는 일	누가
1. 권한 거부 규칙	시크릿 파일과 위험한 명령을 아예 못 읽고 못 돌리게 막는다	개인, 저장소
2. 관리 설정 배포	거부 규칙을 개인이 끌 수 없게 조직 설정으로 내려보낸다	조직
3. 시크릿 스캔 두 군데	커밋 전에 한 번, 에이전트 입출력 경로에서 한 번	저장소, 조직
4. 흑으로 거는 정책	도구 실행 직전에 검사해 규칙에 어긋나면 멈춘다	저장소, 조직
5. 데이터 분류표	넣어도 되는 데이터와 안 되는 데이터를 표 한 장으로 정한다	조직
6. 경유 경로	클라우드 사업자 경유나 내부 게이트웨이로 나가는 길을 하나로 모은다	조직
7. 감사 로그	누가 언제 무엇을 읽게 하고 실행했는지 남긴다	조직

1번은 오늘 저녁에

```
{
  "permissions": {
    "deny": [
      "Read(..env)",
      "Read(..env.*)",
      "Read(..secrets/**)"
    ]
  }
}
```

보안 설정이면서 비용 설정 — 막아 둔 파일은
탐색 비용에서 빠진다

게이트웨이와 감사 로그는 그대로 비용 측정 장치다 — 보안팀과 비용 담당이 한 번에 요청한다

유출 경로와 대책 계층

유출 경로

1. 코드 전송 범위
읽은 파일·명령 출력이 모델 쪽으로
2. 시크릿 파일 읽힘
.env-키 파일을 탐색 중에 읽음
3. 공개 저장소의 MCP 설정
평문 토큰이 커밋돼 공개됨
4. 서드파티 MCP 서버
받은 내용을 어디로 보내는지 모름
5. 프롬프트 인젝션
외부 문서 속 지시를 따름
6. 도구 자체의 유출
배포 과정에서 샘, 공급망 문제

대책 계층 (위: 개인이 오늘, 아래: 조직)

1. 권한 거부 규칙
개인, 저장소
2. 관리 설정 배포
조직
3. 시크릿 스캔 두 군데
저장소, 조직
4. 흑으로 거는 정책
저장소, 조직
5. 데이터 분류표
조직
6. 경유 경로
조직
7. 감사 로그
조직

대책은 한 가지로 끝나지 않는다 — 경로마다 두세 겹으로 막는다

4

이전 방식과 최신 방식 비교 평가 (1/2)

측	이전 방식 (2024~25)	최신 방식 (2026)	평가
과금	정액 좌석제. 요청 횟수 상한만 있고 많이 써도 요금이 같았다	좌석료 위에 토큰이나 크레딧 총량이 얹힌다	비용이 보이게 됐지만 예측은 어려워졌다. 측정하지 않으면 통제가 안 된다
통념	MCP나 플러그인을 붙이면 토큰이 줄어든다고 믿었다. 도구를 많이 붙일수록 유리하다고 봤다	토큰은 컨텍스트에 무엇을 넣느냐로 정해진다. 도구 정의는 지연 로딩하고 CLI를 먼저 쓴다	이전 통념은 정정됐다. 도구는 능력을 넓히는 것이지만 비용을 줄이는 것이 아니다
컨텍스트	한 세션을 하루 종일 열어 두고, 설정 파일에 모든 규칙을 적어 넣었다	작업마다 세션을 비우고, 지시를 붙여 압축하고, 설정은 200줄 아래로 두고 나머지는 스킴으로 뺀다. 탐색은 서브에이전트가 한다	가장 큰 차이다. 긴 세션 하나가 채팅 하루보다 비싸다. 습관만 바뀌어도 절감폭이 가장 크다
캐시	자동이니 신경 쓰지 않았다. 히트율을 볼 방법도 없었다	히트율과 미스 원인을 바로 본다. 앞부분을 보존하고 캐시 수명을 의식한다	캐시 읽기는 정가의 5~10%. 가장 큰 절감 항목인데 미스가 보여야 관리가 된다

4

이전 방식과 최신 방식 비교 평가 (2/2)

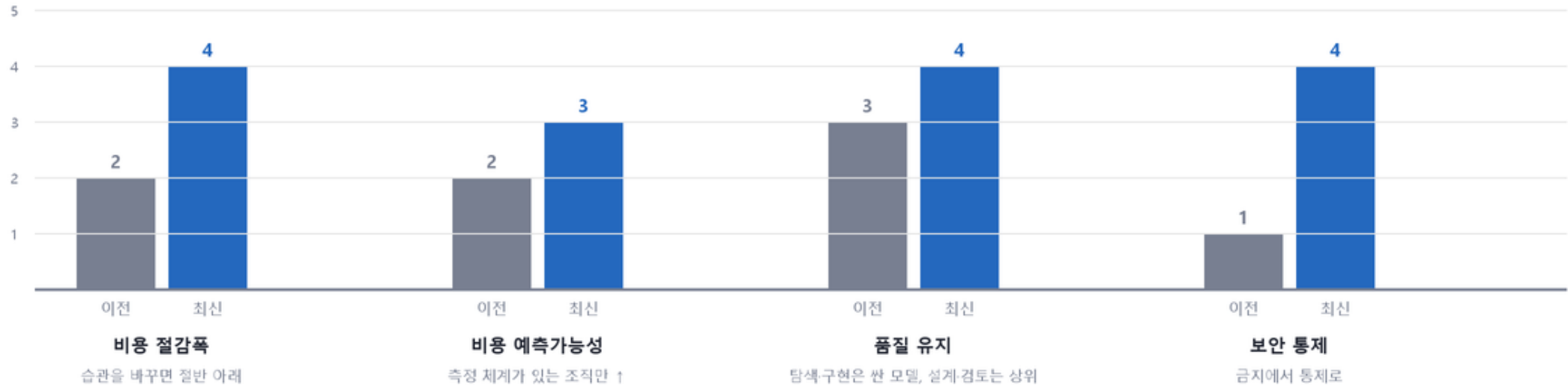
측	이전 방식 (2024~25)	최신 방식 (2026)	평가
모델	최상위 모델을 기본값으로 두고 바꾸지 않았다	탐색, 구현, 설계로 나눠 라우팅하고 서브에이전트마다 모델을 다르게 둔다. 노력 수준으로 사고 토큰을 줄인다	비용을 서너 배 줄일 수 있다. 다만 저가 모델에 90% 이상 몰면 품질이 떨어진다
프롬프트	긴 자연어로 요청하고 바로 실행시킨 뒤 틀리면 다시 요청했다	플랜 모드로 방향을 확정하고 범위와 검증 기준을 같이 준다. 틀어지면 바로 멈춘다	재작업이 가장 큰 숨은 비용이다. 먼저 계획하는 쪽이 싸다
측정	월말 청구서를 보고 알았다. 외부 프로그램 을 깔아야 세부를 볼 수 있었다	프로그램 안에서 세션 비용, 기여도, 캐시 통계, 습관 리포트를 본다. 조직은 OpenTelemetry와 지출 한도를 쓴다	측정이 프로그램 안으로 들어왔다. 세 시간 과정 안에서 전후 시연이 가능해진 이유다
보안	사용을 금지하거나 커밋 전 시크릿 스캔 정도였다. 프로그램 자체는 믿었다	에이전트 입출력 경로에서 스캔하고, 흑으로 행동을 막고, 관리 설정을 배포하고, 내부 게이트웨이를 둔다. 프로그램의 공급망도 의심한다	금지에서 통제로 바뀌었다. AI 보조 커밋의 유출률이 두 배라는 수치가 근거다

과금 줄이 예측가능성은 측정 체계를 갖춘 조직에만 해당된다 — 측정하는 사람만 줄일 수 있다

4

네 축 평가 — 강사 평가이지 측정값이 아니다

이전 방식과 최신 방식, 네 축 평가 (5점 척도, 강사 평가이지 측정값이 아니다)



과금 줄만은 조건부 · 예측가능성은 측정 체계를 갖춘 조직에만 해당 · 그 조건을 빼면 최신 방식이 나머지 축에서 앞선다

오픈웨이트 모델 — 코드가 가는 세 갈래

"오픈웨이트"는 가중치가 공개됐다는 뜻이지, 내 컴퓨터에서 돈다는 뜻이 아니다. 어디서 돌리느냐가 코드가 가는 곳이다



모델 이름보다 추론이 어디서 도느냐를 묻는다 · 그게 코드가 가는 곳이다

Q

중국 · 오픈웨이트 모델 — 싸지만 위험도 같이 산다

모델	입력	출력	비고
Claude Sonnet 5.5	\$2	\$10	비교 기준 · 구현 기본 모델
Claude Haiku 5.5	\$0.10 / 초과 \$0.50	\$0.50 / 초과 \$2.50	가장 싼 Claude · 10/7 출시
DeepSeek Flash	\$0.15 ~ 0.30	\$0.60 ~ 1.20	시간대 요금 · 구독 없음
DeepSeek V4 Pro	\$0.66 ~ 1.32	\$1.98 ~ 3.96	시간대 요금
GLM-5.3 (Z.AI)	\$1.40	\$4.40	코딩 구독 월 \$18부터
Kimi K3	\$3.00	\$15.00	구독 중단 후 개편 중

100만 토큰당 USD · 캐시 미적중 입력 · 2026-10-10 공식 가격표 · DeepSeek는 한국 평일 10~13시, 15~19시 두 배

넣을 데이터	판단
공개 코드 · 시연 샘플	쓸 수 있다 (회사 정책 확인)
사내 코드	사내 서버의 오픈웨이트만, 권한 최소
고객 데이터 · 개인정보 · 시크릿	넣지 않는다

1 데이터가 가는 곳

DeepSeek 처리방침 "서버는 중국". 약관 준거법 중국법. 업체마다 다르다

2 규제 · 사고 이력

2025.1 DB 노출(채팅·API 키). 2025.2 개인정보위 다운로드 중단, 4월 시정권고

3 조직 정책 위반

정부 부처 접속 차단, 카카오·LG유플러스 사내 금지 (2025.2)

4 사내 서버의 한계

NIST 평가: 악성 지시를 따르는 비율 평균 12배 (2025.9). 데이터는 안 나가도 모델 성질은 남는다

Haiku 5.5가 DeepSeek Flash보다 싸다 · 단가가 아니라 작업당 비용으로 비교한다 · 틀려서 다시 시키면 차이가 줄어든다



도구를 바꾼다고 싸지지 않습니다.
측정, 컨텍스트, 티어, 보안 순서로
습관을 바꿔야 싸집니다.

과제 — 오늘 따라 적은 워크시트를 일주일 안에 자기 PC에서 직접 채운다
샘플 저장소로 30분 · 같은 과제 두 개 · 같은 문장 · 새 세션

**회사가 선뜻 밀어주지 못하는 이유는 이해합니다.
그런데 여러분 한 사람에게는
한 달에 100달러입니다.**

이 강의 자료, 같은 구독 하나로 원고·발표자료·시연용 저장소를 만들면서
본업인 PCB 설계 검토, 3D 케이스 설계, 블로그 글까지 같이 돌렸습니다.
회사가 움직여 주기를 기다리지 말고, 자신을 1인 기업이라 생각하고
이 프로그램에 그 돈을 써 보세요.