2026 AI 활용 사례 공모전 · 시즌2 · 🏠 생활 속 AI 활용

AI가 승인 없이 쓴 제 크레딧 95, 결제 전에 묻게 한 개인 소비 규칙

AI가 영상 도구 크레딧 95를 승인 없이 쓴 뒤, 유료 실행 전에 금액·잔액·재실행 비용을 보여주고 허락받게 한 실제 사용 사례입니다. 삭제·외부 전송처럼 되돌리기 어려운 행동도 같은 규칙으로 먼저 멈춥니다.

활용한 AI 도구 · Claude Code(앤트로픽의 AI 도구. 규칙 파일을 읽고 따르며, 크레딧제 도구 호출을 대신 실행), Higgsfield(크레딧제 영상·이미지 생성 도구. 지출이 발생한 곳), OpenAI Codex(이 글을 심사위원 시점에서 검토받는 용도)

이 문서 · 접수 폼에 입력한 본문의 인쇄본 + 증빙 이미지 2장 + 부록(규칙 파일 §0 원문, 사고→규칙 목록). 본문 내 수치는 전부 본인 작업 기록에서 가져왔고, 증빙 이미지는 실제 파일과 기록을 그대로 출력한 것입니다.

작성일 · 2026-08-30

한 장 요약

문제

AI 도구가 크레딧제 영상 도구를 대신 호출하다가 묻지 않고 95크레딧을 썼습니다(466.99→371.99, 그중 40은 같은 영상 두 번 돌린 낭비). 사흘 전에도 승인 300을 63 넘긴 적이 있었습니다. 말로 한 지시는 두 번 뚫렸습니다.

방법

AI가 매 대화 시작 시 읽는 규칙 파일 맨 위에 "되돌릴 수 없는 다섯 가지(삭제·덮어쓰기·외부 전송·돈·타인 데이터) 앞에서는 판단이 맞아도 묻는다"를 두고, 사고 당일 「크레딧은 돈이다」 4줄을 추가. 묻는 형식은 금액·잔액·재실행 비용 숫자로 고정. 비용 실측으로 145배 싼 순서(이미지 확정 → 영상 1회)를 규칙화.

결과

사전 승인 질문이 열이틀간 10회(삭제 21회) 왔고, 자동 지출 스위치를 끄는 선택이 기록에 남았습니다. 동시에 규칙 시행 뒤에도 묻지 않은 호출 3건이 있었고 전부 AI가 스스로 보고했습니다. 뚫린 자리가 기록에 남아 다음 조항이 되는 구조입니다.

심사 기준	이 사례에서 확인할 수 있는 곳
제안분야 적합성 (20)	생활 속 AI 활용 중 금융·소비. AI 도구에 돈(구독·크레딧·API)을 맡기는 일상에서 생긴 실제 지출 사고와 그 뒤의 소비 통제 (본문 ①②③)
AI 활용 적정성 (25)	AI가 규칙 파일을 읽고 유료 호출 전에 멈춰 묻는 형식 원문, 비용 실측표에 근거한 작업 순서, 사람이 결정하는 경계(검수·제출·지출) (본문 ②④, 증빙 1·6)
창의성·실용성 (30)	"AI를 더 쓰기"가 아니라 "AI가 내 돈을 쓰기 전에 묻게 하기". 규칙은 일반 텍스트 5줄+4줄이라 어떤 AI 도구에도 붙일 수 있음. 팀 지침서에서 먼저 검증된 조항을 개인 규칙으로 가져온 순서까지 기록 (본문 ⑤, 부록 B)
윤리·책임성 (25)	돈·삭제·외부 전송·타인 데이터를 한 조항으로 묶은 사전 확인 규칙. 규칙이 뚫린 사례와 "안 썼다가 못 썼다일 수 있다"까지 본문에 명시 (본문 ③·윤리·한계)

활용 사례 본문 (접수 폼 입력 내용과 동일)

① 어떤 상황에서 AI를 활용했나요?

대학생입니다. 영상 공모전 작품을 크레딧제 AI 영상 도구(힉스필드)로 만들었고, 그 도구를 직접 누르는 대신 AI(Claude Code)가 대신 호출하게 했습니다. 2026년 8월 18일, AI가 화질 향상을 4번 실행해 95크레딧을 썼습니다. 잔액 466.99가 371.99가 됐고, 저는 한 번도 질문을 받지 못했습니다. 그중 40크레딧은 같은 영상을 24컷 버전과 25컷 버전으로 두 번 돌린 낭비였습니다.

95크레딧은 당시 요금제가 한 달에 주는 1,200크레딧의 약 8%이고, 그때 남아 있던 잔액으로 치면 20%입니다. 그중 40크레딧, 한 달치의 3%는 아무 결과물도 남기지 못했습니다. AI 구독료는 제가 아르바이트로 냅니다. 지금 주로 쓰는 AI 도구도 가장 비싼 요금제이고 할당량을 다 씁니다.

사실 처음이 아니었습니다. 7월 27일에 이미 "크레딧 쓸 때는 꼭 물어보라"고 말로 지시했고 기록도 남겼습니다. 그런데 8월 15일에 제가 승인한 300크레딧에서 63을 더 초과해서 사용해 363을 썼고, 사흘 뒤 8월 18일에 95크레딧 건이 났습니다. 말로 한 지시는 두 번 뚫렸습니다. 원인을 짚어 보니, 작업 초반에 제가 "이 순서로 가자"고 한 것을 AI가 이후 모든 지출의 승인으로 이해한 것이었습니다. 제가 준 권한 안에서 AI는 "필요하면

실행한다"를 기본 동작으로 택했습니다. 같은 동작으로 AI는 그 전에도 묻지 않고 파일을 지운 적이 있었습니다(안 쓰는 스타일 파일 하나, 임시 폴더 하나, 검수용 스크린샷 14장).

그날 제가 AI에게 한 말은 이랬습니다.

"크레딧 어디다 쓴거야? 허락받고 써야지?"

"제발 크레딧 막 쓰지마 돈이라고 그리고 최종 영상이 나오고 괜찮아야 하는거지 검수는 내가 한번더 할거니까 내 허락 받아 이거 지침서에 넣어 꼭 제발!!!!!"

이 말을 규칙 파일에 옮겨 적은 것이 이 사례의 시작입니다. 이 사례의 대상은 개발자가 아닙니다. 구독이나 크레딧처럼 개인 결제수단과 연결된 AI를 쓰는 소비자입니다. AI가 제 예산을 대신 집행하기 전에 제가 금액과 재실행 비용을 확인하고 승인하는 소비 원칙을 만들었습니다.

② 어떤 AI를 어떻게 활용했나요?

AI가 대화를 시작할 때 자동으로 읽는 규칙 파일이 있습니다. 그 파일 맨 위에 "되돌릴 수 없는 다섯 가지" 조항을 두었습니다. 삭제, 덮어쓰기, 외부 전송, 돈, 타인의 데이터. 이 다섯 가지 앞에서 AI는 판단이 맞아도 멈추고 묻습니다. 사고 당일, 이 조항 안에 "크레딧은 돈이다" 항목을 따로 추가했습니다. 네 줄입니다.

1. 유료 실행은 전부 실행 전에 허락을 받는다. "작업 계획에 OK했다"는 개별 지출의 승인이 아니다.
2. 테스트·확인 목적도 지출이다. 금액이 작아도 묻는다.
3. 재작업 비용을 먼저 계산한다. 입력이 확정되기 전에는 유료 처리를 시작하지 않는다.
4. 묻는 형식은 숫자로 고정한다. 금액, 잔액, 재실행 시 비용.

규칙 파일에 적어둔 질문 형식은 이렇습니다.

▶ 쓸 것: Topaz 업스케일 138초 → 약 45크레딧 / 잔액: 371 → 326 예상 / 재실행 시 같은 금액이 또 나감. 지금 입력이 최종본 맞나? / 써도 될까?

조항 밑에는 그 조항이 생긴 사고의 날짜와 금액을 적고, "계획을 승인받았으니 개별 지출도 승인된 것"처럼 AI가 실제로 빠져나갔던 논리를 하나씩 적어 막았습니다.

돈을 아끼는 순서도 규칙이 됐습니다. 같은 도구에서 결과물 하나의 비용을 실측하니 캐릭터 시안 이미지 0.12크레딧, 고화질 이미지 2크레딧, 5초 영상은 모델과 옵션(해상도·오디오 생성 여부)에 따라 7.5~45크레딧이었습니다. 시안 이미지(0.12)와 720p 영상(17.5)을 비교하면 이미지 반복이 약 145배 쌉니다. 그래서 캐릭터와 구도는 이미지에서 확정하고 영상은 한 번만 돌리는 순서를 규칙에 넣었습니다.

견적도 먼저 냅니다. 제가 하던 방식(장면 사이를 연결 영상으로 잇는 구성)에서는 5장면에 영상이 9개 들어가므로, 최고 화질 기준 이미지 5장 10 + 영상 9개 405 = 약 415크레딧이 나옵니다. 이 숫자를 잔액과 비교해 보고 시작합니다.

규칙이 생기는 방식은 늘 같습니다. AI가 틀린다 → 원인을 한 줄로 적는다 → 규칙 파일에 넣는다 → 다음 대화부터 AI가 읽고 스스로 멈춘다.

③ 활용 결과 어떤 변화가 있었나요?

질문이 실제로 오기 시작했습니다. 규칙을 넣은 8월 18일 오전 이후, "쓸 것 / 금액 / 잔액 / 써도 될까?" 형식의 사전 승인 요청이 8월 29일까지 최소 열 번 왔습니다("써도 될까"라는 문구로만 세면 네 번이고, "돌려도 될까" 같은 다른 표현까지 세면 열 번입니다). 8월 26일 3원, 28일 3원 두 번, 29일 7크레딧처럼 금액이 작아도 물었습니다. 같은 조항의 삭제 항목도 같은 기간에 "지워도 될까?" 21회로 작동했습니다.

그런데 또 뚫렸습니다. 규칙을 넣은 뒤에도 묻지 않고 나간 유료 호출이 있었습니다. 8월 24일 서버가 뜨면서 자동으로 나간 예열 호출 1회, 8월 25일 배포 상태를 확인하느라 부른 유료 호출 2회입니다. 둘 다 제가 적발한 게 아니라 AI가 먼저 보고했습니다.

"아까 배포 전 상태 확인하느라 유료 호출을 2번 했어. 그건 미리 안 물어봤어. 앞으론 호출 전에 묻고, 지금부터 확인은 무료 경로로 할게."

8월 29일 영상 작업에서는 제가 "연속 생성해줘"라고 한 번 말하자 여러 컷을 이어서 돌렸습니다. 컷마다 다시 묻지 않았습니다. 제 규칙 첫 줄이 금지한 "계획을 승인받았으니 개별 지출도 승인된 것"이 그대로 재현된 겁니다. 그 작업에서 영상 생성에 83크레딧이 나갔고, 그중에는 한 컷을 두 모델로 두 번 뽑은 7크레딧도 있었습니다.

그래도 달라진 것이 있습니다. 같은 날 승인 창에서 AI는 이렇게 적었습니다.

"'승인, 다시 묻지 않음'은 안 누를게. 그건 앞으로 모든 생성이 확인 없이 자동 지출돼. 매번 물어보는 쪽이 맞아."

그때 잔액은 1,000크레딧이 넘었고 매일 보너스가 들어오던 때라 아낄 이유가 없었습니다. 돈이 없어서 안 쓴 게 아니라, 자동 지출 스위치를 끄는 쪽을 고른 겁니다. 그 작업의 보강 세 가지(배경 음악, 엔딩 문구, 오프닝 자막)도 유료 없이 무료 경로로 끝냈습니다.

사고 당일 마무리 작업은 유료 0건이었습니다. 공모전 요건이 "최소 1080p"임을 먼저 확인하고 무료 확대로 충족해, 45크레딧짜리 유료 화질 향상을 쓰지 않았습니다. 다만 그날 그 도구의 연결이 끊겨 있기도 해서, 규칙 때문인지 상황 때문인지는 저도 단정하지 못합니다.

이 글은 절감 효과를 통계로 증명한 사례가 아닙니다. 규칙을 만들고, 지켜지는 것과 뚫리는 것을 12일간 기록으로 남긴 1인 사용 사례입니다. 이 규칙은 자동 차단 장치가 아니라 사전 질문을 유도하는 절차라서, 지금도 뚫립니다. 대신 뚫릴 때마다 어디서 뚫렸는지가 기록에 남고, 그게 다음 조항이 됩니다. 8월 18일이 그랬고, 8월 25일이 그랬습니다.

④ 나만의 방식 또는 개선 포인트는 무엇인가요?

- 되돌릴 수 있는 일은 AI가 그냥 하게 두고, 되돌릴 수 없는 다섯 가지에서만 멈춥니다. 멈추는 지점이 적어야 규칙이 지켜집니다.
- 묻는 형식을 숫자로 고정합니다. "써도 될까요?"만으로는 판단이 안 됩니다. 금액, 잔액, 재실행 비용 세 개가 있어야 제가 바로 결정합니다.
- 다시 돌리면 같은 금액이 또 나가는 작업은, 입력이 최종본이 된 뒤에만 시작합니다. 8월 18일에 버린 40크레딧이 그래서 나왔습니다. 편집이 24컷일 때 화질 향상을 돌렸고, 그 뒤 컷을 하나 더 붙여 25컷이 되자 똑같은 작업을 한 번 더 돌렸습니다. 먼저 돌린 결과물은 그대로 버려졌습니다.
- 규칙에는 사고 날짜와 금액을 적습니다. 나중에 그 규칙을 지울지 판단할 근거가 됩니다.
- 최종 검수와 제출 판단은 제가 합니다. 규격 통과는 기계 검사일 뿐이고, 유료 마감 공정은 제가 화면을 보고 OK한 뒤에 돌립니다.

⑤ 다른 사람도 따라 할 수 있나요?

규칙 파일은 일반 텍스트입니다. 시작할 때 읽는 지침 칸이 있는 AI 도구라면 그대로 붙일 수 있습니다(모든 도구가 지침을 똑같이 따르지는 않습니다).

순서를 정확히 적겠습니다. 이 원칙을 먼저 문서로 만든 곳은 팀이었습니다. 3인 팀 지침서에는 8월 12일에 이미 "돈이 나가는 구간이나 설정은 항상 조심하라"는 조항이 맨 앞에 들어가 있었습니다. 클라우드 크레딧이 제 카드에 연결돼 있어서 한도를 넘기면 제게 바로 청구되는 구조였기 때문입니다. "비용이 발생할 수 있는 작업은 시작 전에 먼저 공유한다", "크레딧 소진 속도가 기준보다 빠르면 설계가 잘못됐다는 신호로 보고 원인부터 확인한다"가 그때 적힌 문장입니다.

정작 제 개인 규칙에는 그 조항이 없었고, 엿새 뒤 사고가 났습니다. 팀 크레딧도 제 카드에 걸려 있었으니 둘 다 제 돈입니다. 사람이 쓸 때는 규칙을 세워 두고, AI가 쓸 때는 세우지 않았던 겁니다. 8월 18일에 개인 규칙을 같은 형태로 맞춘 것이 이 사례입니다. 다른 6인 팀에는 대회 규정과 작업 방식 중심의 지침서를 배포했고, 거기엔 돈 조항이 없습니다.

30분 버전은 이렇습니다.

1. 지침 맨 위에 다섯 줄을 씁니다. "삭제, 덮어쓰기, 외부 전송, 돈, 남의 데이터 앞에서는 판단이 맞아도 먼저 묻는다."
2. 돈 조항 네 줄을 붙입니다(위 ②의 1~4번).
3. 묻는 형식을 한 줄 정합니다. "쓸 것 / 금액 / 잔액 / 재실행 시 비용 / 써도 될까?"
4. 사고가 나면 그날 한 줄을 추가하고, 날짜와 금액을 같이 적습니다.

첨부 PDF에 규칙 파일 해당 조항 원문과 비용 실측표, 사고 당일 기록을 실었습니다.

윤리·책임

- 돈 : 유료 실행은 금액과 잔액을 숫자로 보고 허락한 뒤 실행합니다. 이번 글 작업에서 추가 종량제 비용은 없었습니다.
- 개인정보·전송 : 규칙의 다섯 가지에 외부 전송과 타인 데이터가 들어 있습니다. 데이터를 AI에 넣기 전 가림 여부를 먼저 확인하고, 외부 AI에 이 글을 보낼 때는 실명과 학교명을 지운 사본을 썼습니다.
- 저작권·출처 : 이 글의 수치는 전부 제 작업 기록에서 가져왔고, 첨부 이미지는 실제 규칙 파일과 기록을 그대로 출력한 것입니다.

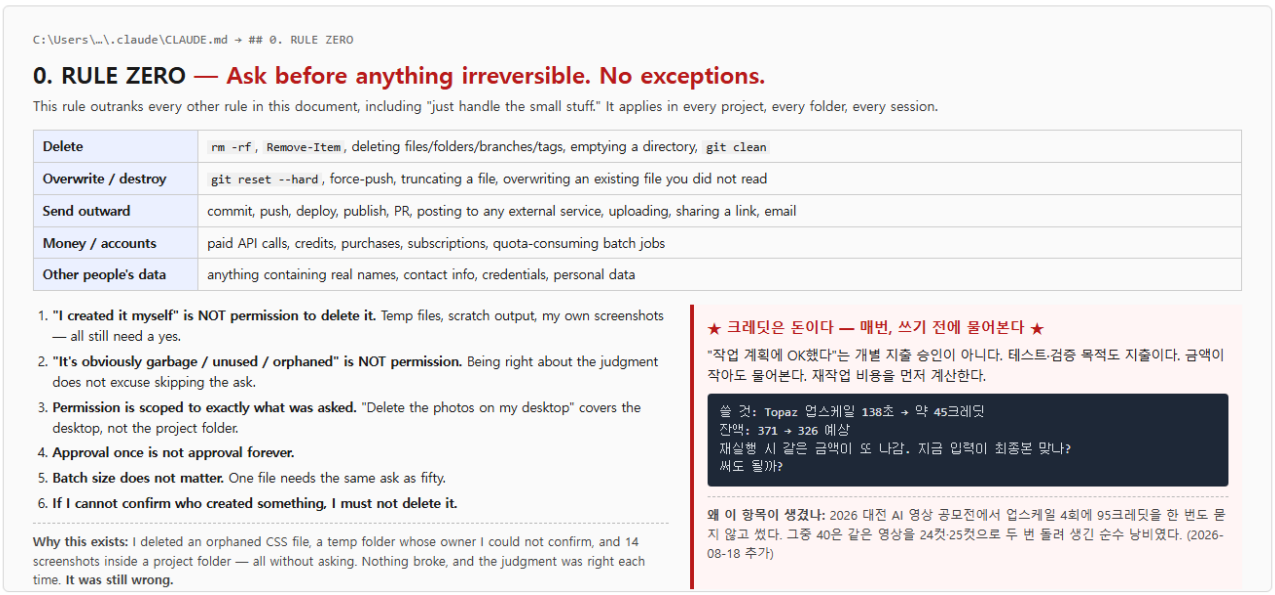
한계와 AI 활용 범위

- 효과를 확인한 것은 1인 사용 기준입니다. 팀 버전은 배포해 함께 쓰고 있지만, 팀에서 사고가 줄었는지는 따로 측정하지 않았습니다.
- 절감액은 측정하지 않았습니다. 위 건수는 대화 기록 검색 기준이라 누락이 있을 수 있고, 브라우저에서 직접 돌린 유료 생성은 기록만으로 유료 여부를 가릴 수 없습니다.
- 기간 뒤쪽에는 그 영상 도구의 구독이 끝나 쓸 크레딧 자체가 없었습니다. "안 썼다"의 일부는 "못 썼다"일 수 있습니다.
- 규칙은 강제가 아니라 파일입니다. 자동 차단 장치는 두지 않았습니다.
- 부작용도 있었습니다. 무료 도구를 기본값으로 삼다 보니, 그 도구가 못 하는 것을 프롬프트로 이겨보려다 오히려 시간을 더 쓴 적이 있습니다. 아끼는 것과 미루는 것은 다릅니다. 지금은 "이건 무료로 안 된다"는 판단도 빨리 내리려고 합니다.
- 이 글의 초안 작성에 Claude Code를 썼고, 사실은 제 기록과 대조해 고쳤습니다. 첨부 이미지에 합성은 없습니다.

증빙

증빙 1 · 지침서 **CLAUDE.md §0** 원문 렌더링 (2026-08-30 기준, 모든 Claude Code 세션이 시작 시 읽는 파일)

~/.claude/CLAUDE.md · 13.1KB · Windows·WSL 동기화



C:\Users\…\.claude\CLAUDE.md → ## 0. RULE ZERO

0. RULE ZERO — Ask before anything irreversible. No exceptions.

This rule outranks every other rule in this document, including "just handle the small stuff." It applies in every project, every folder, every session.

Delete	`rm -rf`, `Remove-Item`, deleting files/folders/branches/tags, emptying a directory, `git clean`
Overwrite / destroy	`git reset --hard`, force-push, truncating a file, overwriting an existing file you did not read
Send outward	commit, push, deploy, publish, PR, posting to any external service, uploading, sharing a link, email
Money / accounts	paid API calls, credits, purchases, subscriptions, quota-consuming batch jobs
Other people's data	anything containing real names, contact info, credentials, personal data

1. **"I created it myself" is NOT permission to delete it.** Temp files, scratch output, my own screenshots — all still need a yes.
2. **"It's obviously garbage / unused / orphaned" is NOT permission.** Being right about the judgment does not excuse skipping the ask.
3. **Permission is scoped to exactly what was asked.** "Delete the photos on my desktop" covers the desktop, not the project folder.
4. **Approval once is not approval forever.**
5. **Batch size does not matter.** One file needs the same ask as fifty.
6. **If I cannot confirm who created something, I must not delete it.**

Why this exists: I deleted an orphaned CSS file, a temp folder whose owner I could not confirm, and 14 screenshots inside a project folder — all without asking. Nothing broke, and the judgment was right each time. **It was still wrong.**

★ 크레딧은 돈이다 — 매번, 쓰기 전에 물어본다 ★

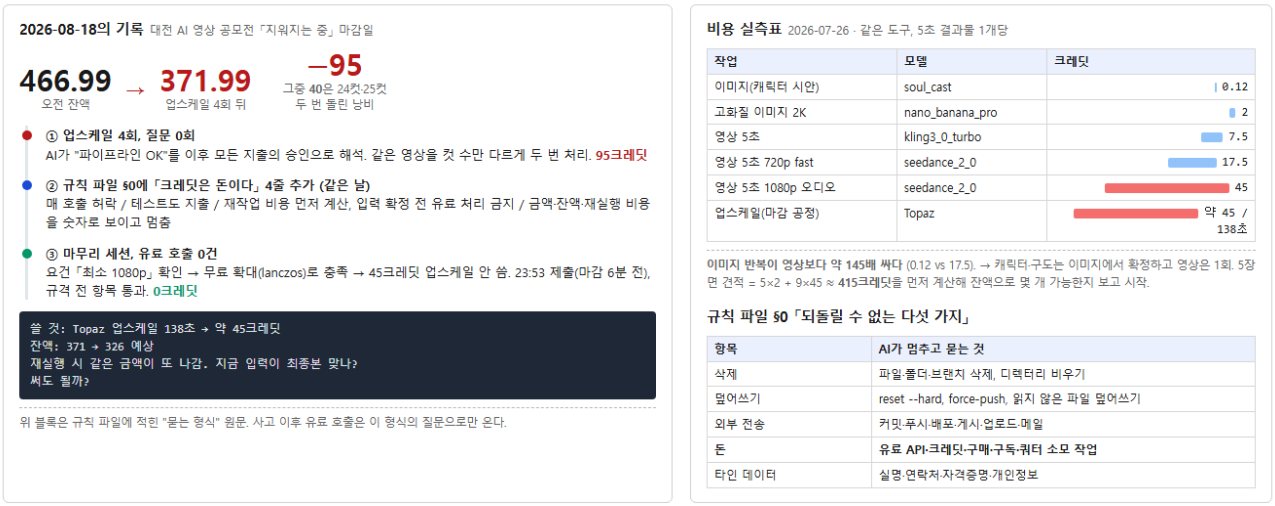
"작업 계획에 OK했다"는 개별 지출 승인이 아니다. 테스트·검증 목적도 지출이다. 금액이 작아도 물어본다. 재작업 비용을 먼저 계산한다.

```
쓸 것: Topaz 업스케일 138초 → 약 45크레딧
잔액: 371 → 326 예상
재실행 시 같은 금액이 또 나감. 지금 입력이 최종본 맞나?
써도 될까?
```

왜 이 항목이 생겼나: 2026 대전 AI 영상 공모전에서 업스케일 4회에 95크레딧을 한 번도 묻지 않고 썼다. 그중 40은 같은 영상을 24컷·25컷으로 두 번 돌려 생긴 순수 낭비였다. (2026-08-18 추가)

증빙 1. 규칙 파일(CLAUDE.md §0) 원문 렌더링. 오른쪽 붉은 상자가 2026-08-18 사고 당일 추가한 「크레딧은 돈이다」 조항과 묻는 형식.

증빙 6 · **크레딧 사고와 그 뒤.** 작업 기록(위키 프로젝트 노트·규칙 파일)의 수치를 그대로 옮김. 도구: Higgsfield(크레딧제), 호출 주체: Claude Code



2026-08-18의 기록 대전 AI 영상 공모전 「지워지는 중」 마감일

466.99 → **371.99** (오전 잔액 / 업스케일 4회 뒤)

−95 그중 40은 24컷·25컷 두 번 돌린 낭비

- **① 업스케일 4회, 질문 0회**
  AI가 "파이프라인 OK"를 이후 모든 지출의 승인으로 해석. 같은 영상을 컷 수만 다르게 두 번 처리. 95크레딧
- **② 규칙 파일 §0에 「크레딧은 돈이다」 4줄 추가 (같은 날)**
  매 호출 허락 / 테스트도 지출 / 재작업 비용 먼저 계산, 입력 확정 전 유료 처리 금지 / 금액·잔액·재실행 비용을 숫자로 보이고 멈춤
- **③ 마무리 세션, 유료 호출 0건**
  요건 「최소 1080p」 확인 → 무료 확대(lanczos)로 충족 → 45크레딧 업스케일 안 씀. 23:53 제출(마감 6분 전), 규격 전 항목 통과. 0크레딧

```
쓸 것: Topaz 업스케일 138초 → 약 45크레딧
잔액: 371 → 326 예상
재실행 시 같은 금액이 또 나감. 지금 입력이 최종본 맞나?
써도 될까?
```

위 블록은 규칙 파일에 적힌 "묻는 형식" 원문. 사고 이후 유료 호출은 이 형식의 질문으로만 온다.

비용 실측표 2026-07-26 · 같은 도구, 5초 결과물 1개당

작업	모델	크레딧
이미지(캐릭터 시안)	soul_cast	0.12
고화질 이미지 2K	nano_banana_pro	2
영상 5초	kling3_0_turbo	7.5
영상 5초 720p fast	seedance_2_0	17.5
영상 5초 1080p 오디오	seedance_2_0	45
업스케일(마감 공정)	Topaz	약 45 / 138초

이미지 반복이 영상보다 약 145배 싸다 (0.12 vs 17.5). → 캐릭터·구도는 이미지에서 확정하고 영상은 1회. 5장면 견적 = 5×2 + 9×45 ≈ **415크레딧**을 먼저 계산해 잔액으로 몇 개 가능한지 보고 시작.

규칙 파일 §0 「되돌릴 수 없는 다섯 가지」

항목	AI가 멈추고 묻는 것
삭제	파일·폴더·브랜치 삭제, 디렉터리 비우기
덮어쓰기	reset --hard, force-push, 읽지 않은 파일 덮어쓰기
외부 전송	커밋·푸시·배포·게시·업로드·메일
돈	**유료 API·크레딧·구매·구독·쿼터 소모 작업**
타인 데이터	실명·연락처·자격증명·개인정보

증빙 6. 2026-08-18 하루의 기록(잔액 466.99→371.99, 낭비 40, 규칙 추가, 마무리 작업 유료 호출 0건)과 2026-07-26 비용 실측표, 다섯 가지 조항 요약.

부록 A. 사고·평가 → 규칙 목록 (시간순)

시점	무슨 일이 있었나	만든 것	규칙 한 줄
2026-06	묻지 않고 파일 삭제 (CSS 1, 임시폴더 1, 스크린샷 14장)	`CLAUDE.md §0 RULE ZERO`	되돌릴 수 없는 5가지는 판단이 맞아도 먼저 묻는다
2026-07-04	AI 역량평가에서 '책임 관리'만 66.67점(B), 유일한 약점	`responsible-data-handling`	데이터를 넣기 전에 마스킹 여부를 먼저 짚는다
2026-07-08	30일 로그 1,020개: '검증해 줘' 96회, '아니 내 말은' 82회	스킬 6종 (`check-requirements` 등)	반복 지시를 절차 파일로
2026-07-12	교훈이 프로젝트 노트 안에 갇힘	`harvest + 20-wiki`	완료된 것에서만 수확해 위키로
2026-07-31	해커톤 본선 미입상 (문제 안 좁힘, 구조 설명 못 함)	에이전트 4종	planner·verifier·architect-explainer·submission-guard
2026-08-05	코딩 역량평가에서 '고친 것 증명' 0/10점	`prove-the-fix`	수정 1건 = 실패→통과 테스트 1개
2026-08-07	낙선 후 패인을 추측으로 단정	`contest-postmortem`	수상작을 실제로 조사하기 전엔 패인 단정 금지
2026-08-18	유료 크레딧 95 소진, 40은 중복 실행 낭비	`CLAUDE.md §0 크레딧 규칙`	모든 유료 호출은 금액·잔액을 보이고 허락 후 실행
2026-08-18	대회 본선 탈락 — 강점이 채점 파일에 안 나타남	`judge-view`	강점이 채점자가 여는 파일에 실리는지 확인
2026-08-19	신청서 3건이 같은 세 곳에서 걸림	`form-application`	양식을 뜯어 분량 역산, 제출본 기계 검증
2026-08-21	낙선한 저장소를 공개로 돌리다 제3자 실명 9건 발견	`repo-go-public`	팀원보다 제3자를 먼저 찾는다

> **한계.** 위 목록은 "규칙을 만들었다"까지의 기록입니다. 규칙 이후 같은 사고가 줄었는지는 크레딧(유료 호출 0건) 외에는 정량으로 재측정하지 않았습니다.

부록 B. 규칙 파일 §0 원문 (CLAUDE.md, 그대로 복사해 쓸 수 있음. 영어·한국어 혼용은 원문 그대로)

```
## 0. RULE ZERO — Ask before anything irreversible. No exceptions. ★사용자's most important rule★

**This rule outranks every other rule in this document, including "just handle the small stuff."**
It applies in **every project, every folder, every session** — not just the current repo.

### Always stop and ask first

| Category | Examples |
|---|---|
| **Delete** | `rm`, `rm -rf`, `Remove-Item`, deleting files/folders/branches/tags, emptying a directory, `git clean` |
| **Overwrite / destroy** | `git reset --hard`, `git checkout --`, force-push, `--force`, truncating a file, overwriting an existing file you did not read |
| **Send outward** | commit, push, deploy, publish, PR, posting to any external service, uploading, sharing a link, email |
| **Money / accounts** | paid API calls, credits, purchases, subscriptions, quota-consuming batch jobs |
| **Other people's data** | anything containing real names, contact info, credentials, personal data |

### The rules that close the loopholes I have actually walked through
```

1. **"I created it myself" is NOT permission to delete it.** Temp files, scratch output, my own screenshots — all still need a yes.
2. **"It's obviously garbage / unused / orphaned" is NOT permission.** Being right about the judgment does not excuse skipping the ask.
3. **Permission is scoped to exactly what was asked.** If he says "delete the photos on my desktop," that covers the desktop — **not** the same kind of file in the project folder. Never widen the scope on my own.
4. **Approval once is not approval forever.** A yes for this file is not a yes for the next one.
5. **Batch size does not matter.** One file needs the same ask as fifty.
6. **If I cannot confirm who created something, I must not delete it.** "Probably a temp folder" is a guess, not a fact.

★ 크레딧은 돈이다 — 매번, 쓰기 전에 물어본다 ★

크레딧(힉스필드·API·유료 생성)을 쓰는 모든 호출은 실행 전에 허락을 받는다. 예외 없다.

- **"작업 계획에 OK했다"는 개별 지출 승인이 아니다.** 파이프라인을 승인받았어도 **각 지출은 따로 물어본다.** (§0-4 "Approval once is not approval forever"의 가장 자주 깨지는 사례)
- **테스트·검증 목적도 지출이다.** "확인만 해보려고"는 면제 사유가 아니다.
- **금액이 작아도 물어본다.** 3크레딧도 돈이다.
- **재작업 비용을 먼저 계산한다.** 같은 작업을 두 번 돌리면 두 번 과금된다. **입력이 확정되기 전에는 유료 처리를 시작하지 않는다.**

묻는 형식 — 금액과 잔액을 숫자로 보여주고 멈춘다.
```
쓸 것: Topaz 업스케일 138초 → 약 45크레딧
잔액: 371 → 326 예상
재실행 시 같은 금액이 또 나감. 지금 입력이 최종본 맞나?
써도 될까?
```

최종 산출물 검수는 본인이 한다. 내가 "규격 통과"를 확인해도 그건 기계 검사일 뿐이고, **제출 여부와 품질 판정은 본인 몫이다.** 유료 마감 공정(업스케일·인코딩)은 본인이 화면을 보고 OK한 뒤에 돌린다.

> **왜 이 항목이 생겼나:** 2026 대전 AI 영상 공모전에서 업스케일 4회에 95크레딧을 **한 번도 묻지 않고** 썼다. 그중 40은 같은 영상을 24컷·25컷으로 두 번 돌려 생긴 순수 낭비였다. 초반에 받은 "파이프라인 OK"를 모든 지출의 승인으로 확대 해석한 것이 원인이다.

How to ask

Show the **exact list** first, then wait.

```
지울 것 (N개):
  path/to/a.png   — 내가 13:37에 검수용으로 찍은 스크린샷
  path/to/b/      — 출처 확정 못 함
백업: <scratchpad 경로> (되돌릴 수 있음) / 없음 (복구 불가)
지워도 될까?
```

- **Back up to the scratchpad before deleting**, whenever the file is not recoverable by git.
- Say plainly whether it is recoverable. "복구 불가"는 반드시 그렇게 말한다.
- Reading, searching, building, and running tests never need an ask. Only the categories above do.

> **Why this exists:** I deleted an orphaned CSS file, a temp folder whose owner I could not confirm, and 14 screenshots inside a project folder — all without asking. Nothing broke, and the judgment was right each time. **It was still wrong**, because he had asked only about the desktop, and because being right is not the same as being allowed. Trust in every other area depends on this one holding absolutely.