fable-mode

Opus 4.8을 Fable 5 행동 규범으로 굴린다

Fable 세션에만 주입되는 시스템 프롬프트 층을 훅 4종으로 이식했다. **행동은 97%까지 따라오고, 비용은 Fable의 72%.** 웨이트가 만드는 추론 갭은 남는다 — 그 경계까지 실측했다.

실측 — **3조건 × 4과제** · 트랜스크립트 usage 집계 · 6차원 루브릭 채점

지동석 · jidonglab jd@jidonglab.com v1.4 · github.com/jee599/fable-mode-kit

— 스택 구성

훅 4종 detect · context · subagent · stop

자동감지, 매 턴 규범 주입, 서브에이전트 커버, 자가검증·누설가드

output style + 래퍼 fable-like · claude-fablelike

시스템 프롬프트 레벨 이식 + xhigh·ultracode 원샷 런처

토글 + 텔레메트리 /fable-mode · stats

on/off/status — 주입 횟수·토큰 오버헤드 세션별 기록

설치는 플러그인 2줄 — **Opus 세션 자동 감지**

행동 충실도 (36점 루브릭)	비용 (동일 4과제 합)	산출물 등가성 (스택↔Fable)	v1.4 주입 오버헤드
97% 순정 64%	Fable의 0.72×	80–95%	minor 턴 −84%

Fable과 Opus의 차이는 두 층 — 한 층만 이식된다

2026-07-02 리서치(양 모델 헤드리스 프로브 + CLI 바이너리 분석)로 확인 — 하네스(effort·워크플로·메모리·서브에이전트)는 두 모델 공통이고, 다른 건 **시스템 프롬프트와 웨이트**뿐이다.

Layer 1 · 이식 가능

시스템 프롬프트 행동 층

Fable 세션에만 주입되는 소통·자율실행·검증 규범(실측 확인). 텍스트이므로 훅으로 재현할 수 있다 — 결론 우선, 턴 완결성, 허락 질문 금지, 검증 후 보고, 스코프 준수.

→ 이 킷이 이식하는 전부

Layer 2 · 이식 불가

모델 웨이트

얽힌 단일 패스 추론, 장기 자율 런 유지력. 3rd-party 벤치 기준 5–11pt(추정)의 격차 — 프롬프트로는 닫히지 않는다.

→ Fable 직행 또는 다중 검증으로 보정

단가 — Fable $10/$50, Opus 4.8 $5/$25 (Mtok 입력/출력, 정확히 2×). 킷의 명제: 2× 단가를 내지 않고 Layer 1을 산다.

Method · 4-Hook Stack

훅 4종이 세션 수명주기 전체를 잡는다

감지, 매 턴 규범, 서브에이전트, 종료 검증 — 셸 스크립트 4개에 상태는 **마커 파일**뿐. 데몬도 프록시도 없다.

SessionStart

fable-detect.sh

- Opus 모델 자동 감지 — 입력·조상 프로세스 플래그·settings 3단 폴백
- 세션 마커 기록, Fable 세션은 해제

UserPromptSubmit

fable-context.sh

- 매 턴 규범 재주입 — 장기 컨텍스트 희석 방지
- v1.4 적응형: major는 풀 블록, minor는 리마인더
- major 턴 마킹 + 주입 텔레메트리

SubagentStart

fable-subagent.sh

- 빌트인·커스텀·워크플로 서브에이전트 전부에 중립 conduct 블록
- 정의에 이미 있으면 자동 스킵 — 이중 주입 방지

Stop

fable-stop-verify.sh

- major 턴 1회 자가검증 — 검증 후 보고·결론 우선·스코프
- v1.4 누설가드: 내부 용어 감지 시 1회 재작성

활성화 3경로(자동감지 · FABLE_MODE=1 · /fable-mode on) + 킬스위치 FABLE_MODE=0. Fable 세션은 트랜스크립트 감지로 자동 침묵 — 이중 주입 없음.

— Methodology · A/B/C

같은 프롬프트, 같은 조건 — 다른 건 스택뿐

헤드리스 CLI로 조건당 동일 과제를 실행하고, 트랜스크립트의 usage 필드로 토큰을, 공식 단가로 비용을 집계했다. 행동은 **6차원 루브릭 36점**으로 심판 채점.

3조건 — 동일 프롬프트 · 동일 effort · empty MCP

- A — 순정 Opus 4.8 FABLE_MODE=0, 스택 완전 비활성
- B — Opus 4.8 + fable-mode 훅 4종 활성 — 측정 대상
- C — Fable 5 원본 기준선. 스택은 자동 침묵

4과제 유형 — 실사용 카테고리

- 진단 버그 원인 질문 — 스코프 준수(무단 수정 금지)를 본다
- 구현 slugify 함수 — 실행 검증·근거 보고를 본다
- 모호한 정리 "로그 좀 정리해줘" — 자율 실행 vs 역질문 정지를 본다
- 단순 질문 rebase vs merge — 응답 형태·오버헤드를 본다

진단·구현·단순 질문은 2026-07-06~07 v1.4 스택 재실측, 모호한 정리는 2026-07-03 실측. 공용 훅(omc-dial 등)은 세 조건에 동일 적용 — 조건 간 비교에 중립.

Results · Conduct Fidelity

행동 충실도 64% → 97% — 격차 대부분이 닫힌다

과제 3종 × 6차원 루브릭 36점 만점 — 결론 우선 · 형태 적합 · 스코프 · 자율 실행 · 검증 보고 · 완결성.

B의 35점은 v1.3 누설 회귀(자가검증 문구가 출력에 새는 문제)를 수정한 뒤 재실행으로 확인한 값 — 남은 1점은 표현 밀도 차이다.

Results · What Actually Got Fixed

순정 Opus의 대실패 2건이 완전 교정됐다

점수 차의 대부분은 미세한 톤이 아니라 **명백한 행동 실패**에서 왔다 — 그리고 그 실패는 규범 주입으로 사라졌다.

순정 Opus 4.8	+ fable-mode
"왜 버그가 나?" 진단 요청에 무단으로 파일을 수정하고, 실행 없이 결과를 단정	진단만 보고하고 멈춤 — 원인은 **실행으로 검증** (6/12 → 12/12)
모호한 "정리해줘" 요청에 조사만 하고 "어떤 방식을 원하세요?" 정지	비파괴 디폴트를 **즉시 실행** , 파괴적 단계만 보류 (5/12 → 12/12)
v1.3 스택 자체 회귀 — 자가검증 문구가 사용자 출력에 누설	v1.4 누설가드 — 최종 메시지를 **결정론적으로 감지** 해 1회 재작성 강제

동일 4과제 합계, 스택은 Fable의 72% 비용

트랜스크립트 usage × 공식 단가(캐시 읽기·쓰기 포함). Fable은 출력 토큰을 절반만 쓰지만 — **단가 2×가 역전**시킨다.

과제	A · 순정 Opus $5 / $25 Mtok	B · Opus + 스택 $5 / $25 Mtok	C · Fable 5 $10 / $50 Mtok
진단 버그 원인 설명 · 7/7 재실측	$0.23	$0.28	$0.49
구현 slugify + 검증 · 7/7 재실측	$0.34	$0.45	$0.49
모호한 정리 로그 파일 정리 · 7/3 실측	$0.86	$1.13	$1.52
단순 질문 rebase vs merge · 7/6 실측	$0.12	$0.13	$0.27
합계 · Fable 대비	**$1.54 · 0.56×**	**$1.99 · 0.72×**	**$2.77 · 1.00×**

출력 토큰 합 — A 12.8k · B 20.0k · C 9.8k. 스택의 추가분(A→B +$0.45)은 규범 주입과 자가검증 턴의 값이고, Fable은 토큰을 절반만 쓰지만 단가 2×가 역전시킨다. 과제별 편차는 크다 — 진단은 0.57×, 구현은 0.91×.

Results · Output Equivalence

산출물 등가성 80–95% — 결론은 같고 디폴트 선택만 갈린다

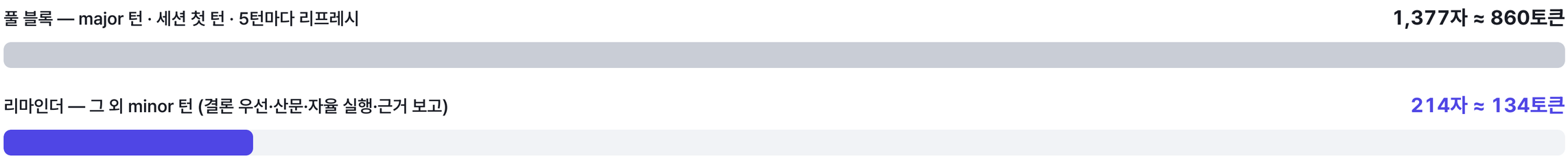
과제별로 스택(B)과 Fable(C)의 최종 산출물을 나란히 비교 — 기능 등가성·응답 형태·행동 규범 기준.

1	**진단 — 동일 결론, 동일 행동** 7/3(가변 기본 인자)과 7/7 재실측(키 불일치) 모두 — 같은 원인 지목, 수정 없이 진단만, 실행 근거 제시까지 동일	등가성 ~95%
2	**구현 — 같은 스펙 충족, 같은 검증 습관** 요구 4조건을 모두 만족하는 slugify + 실행 검증. 7/7 재실측 산출물 3벌을 같은 케이스로 돌리면 출력이 전부 일치	등가성 ~90%
3	**단순 질문 — 결론 우선 산문, 같은 내용** 첫 문장에 답(merge commit 보존 vs 히스토리 재작성), 공유 브랜치 경고까지 동일 — 문체 밀도만 차이	등가성 ~85%
4	**모호한 정리 — 같은 규범, 다른 디폴트** 둘 다 비파괴 실행 후 파괴 작업만 보류. B는 logs/ 폴더 정리, C는 gzip 아카이브 — 합리적 디폴트가 갈렸을 뿐	등가성 ~80%

v1.4 · Adaptive Injection

주입을 턴 크기에 맞춘다 — minor 턴 −84%

규범 블록은 매 턴 들어가야 효과가 유지되지만, 짧은 질답 턴에 1,377자 전체는 낭비였다. v1.4는 **major 턴·세션 첫 턴·5턴마다**만 풀 블록을 넣고, 나머지는 핵심 리마인더로 줄인다.



- **50턴 세션 주입 오버헤드 (계산치)**

  minor 턴 비중 50%면 **−34%**, 70%면 −47%, 80%면 −54%. 매 주입은 stats/<세션id>.tsv에 기록되고, /fable-mode status가 세션별 오버헤드를 보고한다.

모델을 믿는 대신 결정론으로 지킨다

v1.4의 나머지 고도화 — 전부 결정론적이거나 관측 가능한 장치다.

누설가드 — 신뢰가 아니라 감지

턴 종료 시 최종 메시지를 스캔해 내부 지침 용어('자가검증', 'fable-mode' 등)가 새어 나오면 1회 재작성을 강제한다. 마커로 무한 루프를 차단 — 7/3 E2E가 잡은 유일한 회귀 클래스를 결정론으로 봉인.

플러그인 dedupe — 이중 주입 방지 확대

서브에이전트 정의에 conduct 블록이 이미 있으면 스킵하는 검사를 플러그인 경로(marketplaces·cache, 네임스페이스 타입)까지 확장했다.

텔레메트리 — 오버헤드를 숫자로

주입마다 타입·문자수를 세션별 tsv에 기록한다. /fable-mode status가 full/lite 횟수와 토큰 환산치를 보고 — 이 보고서의 오버헤드 수치도 여기서 나왔다.

major 동사 확장 — 자가검증 커버리지

커밋·푸시·배포·발행·보고서·테스트·실행·고도화·최적화 등을 추가했다. 짧은 영어 동사(run·test·make)는 부분 문자열 오탐 때문에 의도적으로 제외 — 80자 규칙이 받친다.

— Verification

분기 테스트 14/14 · 헤드리스 3런 · 커밋 173f008 푸시

모든 분기를 스크립트로 직접 두드려 확인한 뒤 커밋했다 — 로컬 설치본과 킷 레포는 diff 0으로 동기.

fable-context.sh — 주입 분기

- 킬스위치 FABLE_MODE=0이면 침묵 통과
- 세션 첫 주입은 풀 블록 통과
- minor 턴은 리마인더 + stats 기록 통과
- major 턴은 풀 블록 + 마커 통과
- 5턴마다 풀 블록 리프레시 통과

stop-verify · subagent · detect · E2E

- 자가검증은 턴당 1회, 재정지 시 침묵 통과 — 루프 없음
- 누설 텍스트 감지 시 1회 재작성 블록 통과
- Fable 트랜스크립트면 자동 침묵 통과
- conduct 마커 보유 에이전트는 주입 스킵 통과
- 헤드리스 실주입 — 7/6 질문 과제 3런 통과

배포 — github.com/jee599/fable-mode-kit 커밋 173f008(v1.4) 푸시 완료. 플러그인 2줄 설치와 클래식 인스톨러 양쪽 유지.

— Honest Limits

97%는 행동 점수다 — 지능 벤치마크가 아니다

이 보고서가 주장하지 않는 것들을 명확히 남긴다.

표본 — 4과제 × 조건당 1런

과제는 행동 차이가 드러나도록 설계됐다. 통계적 신뢰구간을 말할 표본이 아니며, 재현용 러너는 보존되어 있다.

웨이트 갭 — 얽힌 추론·장기 자율 런은 남는다

3rd-party 벤치 5–11pt(추정). 고위험 결정·장기 자율 작업은 Fable 직행이 맞고, 스택으로 가려면 팬아웃과 적대 검증으로 보정한다.

부분 재실측 — 모호 정리만 7/3 값 유지

한도 리셋 후 재실행으로 진단·구현은 v1.4 스택 재실측이 표에 반영됐다. 모호 정리는 A·B가 다음 세션 한도에 다시 걸려 7/3 값을 유지 — 러너는 보존되어 있다.

환경 — 공용 훅이 세 조건에 공통 적용

omc-dial 등 로컬 공용 훅이 A·B·C 모두에 걸렸다. 조건 간 비교는 중립이지만, A의 절대값은 '완전 민낯 Opus'보다 다소 유리할 수 있다.

— 결론

행동의 97%를 절반 단가로 — 경계만 지키면 남는 장사

라우팅 규칙은 세 줄이면 된다.

01 · 기본값

대화형 작업은 스택

구현·진단·문서·리뷰 등 행동 규범이 품질의 대부분인 일. Fable 대비 0.72× 비용에 등가성 80–95%.

→ Opus 세션 자동 감지로 켜진다

02 · 예외

얽힌 추론은 Fable 직행

단일 패스에 걸린 고난도 추론, 장기 자율 런, 고위험 결정. 웨이트 갭은 프롬프트로 닫히지 않는다.

→ 또는 팬아웃 + 적대 검증 보정

03 · 옵트아웃

크론·원샷은 FABLE_MODE=0

자가검증이 턴을 하나 더 쓴다 — 타임아웃 예산이 있는 헤드리스·크론은 끄는 게 맞다.

→ 킬스위치가 모든 경로를 이긴다

출처 — 2026-07-03 E2E(3과제 × 3조건) + 2026-07-06~07 재실측(진단·구현·질문 × 3조건)·분기 테스트 14건, 트랜스크립트 usage 실측 · 단가는 공식 문서 기준 · 벤치 갭은 3rd-party 추정. github.com/jee599/fable-mode-kit