

클로드코드 규칙 관리 · 무료

클로드코드 규칙 파일 팩

CLAUDE.md 를 쪼개고, 경로별로 붙이고, 프로젝트끼리 공유하는 법
.claude/rules 시작 템플릿 · paths 글롭 표 · 심볼릭 링크 공유 · 200줄 다이어트

바이브 메이커

docs.vibemake.kr

CONTENTS

목차

PART 0 **이 팩이 푸는 문제** 1

0.1 이 팩에 든 것 1

0.2 먼저 알아둘 두 가지 1

PART 1 **시작 템플릿 3종** 3

1.1 api.md — 경로를 건 규칙 3

1.2 testing.md — 테스트 파일 전용 4

1.3 security.md — 항상 붙는 규칙 4

PART 2 **paths 글롭 패턴 표** 5

2.1 기본 패턴 5

2.2 여러 개 걸기 · 중괄호 5

2.3 자주 틀리는 함정 5

2.4 언제 붙는가 6

PART 3 **여러 프로젝트가 규칙을 같이 쓰기** 7

3.1 심볼릭 링크 7

3.2 실제 운영 모양 7

3.3 개인 규칙은 링크도 필요 없다 8

3.4 주의 8

PART 4 **CLAUDE.md 200줄 다이어트** 9

4.1 어디로 내릴지 판단표 9

4.2 옮겨도 안 줄어드는 것 9

4.3 남기는 문장 쓰는 법 10

4.4 다이어트 절차 10

4.5 마지막 확인 10

이 자료의 모든 내용은 **클로드코드 공식 문서(code.claude.com/docs)** 원문과 1:1 대조해 정리했습니다. 회색 코드 상자는 그대로 복사해 쓰시면 됩니다.

PART 0 전체 5장

이 팩이 푸는 문제

클로드가 규칙을 자꾸 무시한다면, 실력이 아니라 **파일 길이** 문제일 수 있습니다.

`CLAUDE.md` 는 대화를 시작할 때마다 **통째로** 들어갑니다. 공식 문서의 표현은 이렇습니다.

> 💡 "CLAUDE.md files are loaded into the context window at the start of every session."

그리고 길어질수록 지켜질 확률이 떨어집니다.

> 💡 "Longer files consume more context and reduce adherence."

그래서 답은 "더 강하게 쓰기"가 아니라 **쪼개서 필요할 때만 붙이기** 입니다.

0.1. 이 팩에 든 것

장	내용
PART 1	`.claude/rules/` 시작 템플릿 3종 (그대로 복사)
PART 2	`paths` 글롭 패턴 표 + 자주 틀리는 함정
PART 3	**심볼릭 링크로 여러 프로젝트가 규칙을 같이 쓰기**
PART 4	CLAUDE.md 200줄 다이어트 판단표

0.2. 먼저 알아둘 두 가지

하나. 규칙은 강제가 아니라 맥락입니다.

> 💡 "Claude treats them as context, not enforced configuration."

무조건 막아야 하는 게 있으면 규칙이 아니라 **훅(PreToolUse)** 으로 거세요. 규칙은 "이렇게 해줘"이지 "이건 못 해"가 아닙니다.

둘. `paths` 없는 규칙은 항상 로드됩니다.

경로를 안 걸면 조건부가 아니라 **무조건**입니다. 이 차이를 모르면 쪼개도 컨텍스트가 안 줄어듭니다.

PART 1 전체 5장

시작 템플릿 3종

폴더만 만들면 끝입니다. 새로 깔 것 없습니다.

```
your-project/
├── CLAUDE.md            ← 항상 로드되는 짧은 핵심
└── .claude/
    └── rules/
        ├── testing.md   ← 테스트 건드릴 때만
        ├── api.md       ← API 코드 건드릴 때만
        └── security.md  ← 항상 (paths 없음)
```

1.1. api.md — 경로를 건 규칙

MARKDOWN

```
---
paths:
  - "src/api/**/*.ts"
---

# API 규칙

- 모든 엔드포인트는 입력값 검증을 먼저 한다
- 에러 응답은 `{ code, message }` 형태로 통일
- 새 엔드포인트는 OpenAPI 주석을 함께 단다
```

맨 위 세 줄이 전부입니다. `src/api/` 아래 `.ts` 파일을 건드릴 때만 이 규칙이 붙습니다.

1.2. testing.md — 테스트 파일 전용

MARKDOWN

```
---
paths:
  - "**/*.test.ts"
  - "**/*.spec.ts"
  - "tests/**/*"
---

# 테스트 규칙

- 테스트는 행동을 검증한다. 구현 세부를 검증하지 않는다
- 새 버그를 고치면 그 버그를 재현하는 테스트를 먼저 추가한다
- `expect` 없는 테스트는 만들지 않는다
```

1.3. security.md — 항상 붙는 규칙

MARKDOWN

```
# 보안 규칙

- 비밀키·토큰·비밀번호를 코드나 로그에 남기지 않는다
- `.env*` 파일은 커밋하지 않는다
- 외부 입력은 신뢰하지 않는다
```

`paths` 가 **없습니다.** 그래서 항상 로드됩니다.

⚡ 한 단계 더 규칙 파일은 주제 하나에 하나씩. 파일명이 곧 목차입니다. `rules.md` 같은 이름을 만들면 결국 두 번째 CLAUDE.md 가 됩니다.

PART 2 전체 5장

paths 글롭 패턴 표

`paths` 는 "이 규칙을 언제 붙일지"를 정하는 유일한 스위치입니다.

2.1. 기본 패턴

패턴	걸리는 것
`**/*.ts`	모든 디렉토리의 TypeScript 파일
`src/**/*`	`src/` 아래 전부
`*.md`	**프로젝트 루트의** 마크다운만
`src/components/*.tsx`	그 폴더 바로 아래만 (하위 폴더 제외)

`*` 는 한 단계, `**` 는 여러 단계입니다. `src/*/x.ts` 와 `src/**/x.ts` 는 다릅니다.

2.2. 여러 개 걸기 · 중괄호

MARKDOWN

```
---
paths:
  - "src/**/*.{ts,tsx}"
  - "lib/**/*.ts"
  - "tests/**/*.test.ts"
---
```

중괄호는 곱해집니다. `src/*.{ts,tsx}` 는 2개로, `{a,b}/{c,d}/*.{ts,tsx}` 는 8개로 펼쳐집니다.

한 규칙의 `paths` 전체가 **1,000개 · 4MiB** 예산을 함께 씁니다. 넘으면 그 패턴은 펼쳐지지 않은 채로 쓰이고, 중괄호가 글자 그대로 취급돼 **아무 파일도 안 걸립니다.**

2.3. 자주 틀리는 함정

대괄호 `[` 는 문자 클래스입니다.

`[abc]` 처럼 읽히지 않는 `[` 가 들어가면 그 패턴은 **아무것도 매칭하지 않습니다.**

```markdown
# ❌ photos [2024/**        ← 매칭 0
# ✅ photos \[2024/**       ← 이스케이프
```

같은 규칙의 다른 패턴들은 계속 동작하니, 조용히 한 줄만 죽습니다. 그래서 더 위험합니다.

경로를 안 걸면 조건부가 아닙니다.

`paths` 없는 규칙은 항상 로드됩니다. "쪼갰는데 왜 컨텍스트가 그대로지?" 의 대부분이 이겁니다.

2.4. 언제 붙는가

> 💡 "Path-scoped rules trigger when Claude reads files matching the pattern, not on every tool use."

클로드가 그 파일을 읽을 때 붙습니다. 도구를 쓸 때마다가 아닙니다.

그래서 `/compact` 뒤에는 경로 규칙이 자동으로 다시 안 들어옵니다. 해당 파일을 다시 만질 때 다시 붙습니다.

> ⚡ 한 단계 더 규칙이 안 먹는 것 같으면 `/context` 로 뭐가 실제로 로드됐는지 먼저 보세요. 규칙을 더 세게 쓰는 건 그다음입니다.

PART 3 전체 5장

여러 프로젝트가 규칙을 같이 쓰기

여기가 판을 바꾸는 부분입니다.

규칙 한 벌을 만들어 두고 **여러 저장소가 같은 걸 바라보게** 할 수 있습니다. 복사가 아니라 링크입니다. 한 곳을 고치면 전부 바뀝니다.

3.1. 심볼릭 링크

```bash
# 폴더 통째로 공유
ln -s ~/shared-claude-rules .claude/rules/shared

# 파일 하나만 공유
ln -s ~/company-standards/security.md .claude/rules/security.md
```

공식 문서가 지원한다고 명시한 방식입니다.

> 💡 "The `.claude/rules/` directory supports symlinks, so you can maintain a shared set of rules and link them into multiple projects."

링크는 따라가서 정상 로드되고, 순환 링크도 안전하게 처리됩니다.

3.2. 실제 운영 모양

```
~/shared-claude-rules/
├── security.md       ← paths 없음 = 어느 프로젝트에서나 항상
├── commit.md         ← 커밋 메시지 규칙
└── review.md         ← 리뷰 기준

project-a/.claude/rules/shared → ~/shared-claude-rules
project-b/.claude/rules/shared → ~/shared-claude-rules
```

새 프로젝트를 열 때 `ln -s` 한 줄이면 팀 표준이 그대로 따라옵니다.

3.3. 개인 규칙은 링크도 필요 없다

`~/.claude/rules/` 에 둔 규칙은 **이 컴퓨터의 모든 프로젝트**에 적용됩니다.

```
~/.claude/rules/
├── preferences.md    ← 내 코딩 취향
└── workflows.md      ← 내 작업 방식
```

사용자 규칙이 먼저 로드되고 프로젝트 규칙이 나중에 로드됩니다. 즉 **프로젝트 규칙이 우선**입니다.

3.4. 주의

- 공유 규칙에 **프로젝트 고유 사정**을 넣지 마세요. 다른 저장소에서 헛발질합니다
- 공유 폴더 안 규칙에 `paths` 를 걸 때는 **모든 프로젝트가 같은 구조**인지 확인하세요
- 팀 저장소에 남의 규칙이 딸려 오면 `claudeMdExcludes` 로 걸러낼 수 있습니다

> ⚡ 한 단계 더 공유 규칙은 git 저장소로 만들어 두면 이력이 남습니다. "왜 이 규칙이 생겼지"를 나중에 추적할 수 있습니다.

PART 4 전체 5장

CLAUDE.md 200줄 다이어트

공식 문서가 말하는 상한은 이겁니다.

> 💡 "target under 200 lines per CLAUDE.md file."

지금 몇 줄인지부터 세보세요.

```bash
wc -l CLAUDE.md .claude/CLAUDE.md 2>/dev/null
```

4.1. 어디로 내릴지 판단표

한 줄씩 보면서 이 표로 분류합니다.

이 내용이...	어디로
매 세션 항상 필요 (빌드 명령·프로젝트 구조·전역 컨벤션)	**CLAUDE.md 에 남긴다**
특정 폴더·확장자 작업할 때만 필요	`.claude/rules/` + `paths`
여러 단계짜리 절차 (배포 순서·릴리스 체크리스트)	**스킬**(`/skills`)
무조건 막아야 하는 것 (커밋 금지·명령 금지)	**훅**(PreToolUse)
여러 저장소가 공유	**심볼릭 링크** (PART 3)

공식 문서의 기준도 같습니다.

> 💡 "If an entry is a multi-step procedure or only matters for one part of the codebase, move it to a skill or a path-scoped rule instead."

4.2. 옮겨도 안 줄어드는 것

`@path/to/file` **import 는 컨텍스트를 줄이지 않습니다.** 정리는 되지만 시작할 때 같이 로드됩니다.

> 💡 "Splitting into @path imports helps organization but doesn't reduce context, since imported files load at launch."

줄이려면 `paths` 로 **조건부**로 만들어야 합니다.

4.3. 남기는 문장 쓰는 법

검증 가능하게 씁니다. 애매하면 안 지켜집니다.

❌	✅
포맷 잘 맞춰줘	들여쓰기는 스페이스 2칸
테스트 잘 해	커밋 전에 `npm test` 를 돌린다
파일 정리해	API 핸들러는 `src/api/handlers/` 에 둔다

그리고 **서로 모순되는 규칙**을 없애세요. 두 규칙이 부딪히면 클로드가 임의로 하나를 고릅니다.

4.4. 다이어트 절차

1. `wc -l` 로 현재 줄 수 확인
2. 위 판단표로 각 줄을 분류
3. 폴더 한정 내용을 `.claude/rules/*.md` 로 내리고 `paths` 를 건다
4. 절차성 내용은 스킬로 뺀다
5. 다시 `wc -l` — 200줄 아래인지 확인
6. `/context` 로 실제 로드 상태 확인

4.5. 마지막 확인

- [ ] CLAUDE.md 200줄 미만
- [ ] 규칙 파일마다 주제 하나
- [ ] 폴더 한정 규칙에 `paths` 가 걸려 있다
- [ ] `paths` 없는 규칙은 "정말 항상 필요한 것"만
- [ ] 모순되는 규칙 없음
- [ ] 공유할 규칙은 심볼릭 링크로

이 팩의 모든 내용은 **클로드코드 공식 문서** 원문과 1:1 대조해 정리했습니다. 원문: `code.claude.com/docs/en/memory`

바이브 트렌드 · @vibe.trender