KAIROS AI STUDIO · 무료 요약집

cc-tips · EP.01 → EP.10

Claude Code 첫걸음 10편 핵심 요약

카드엔 못 담은 '왜·언제'까지 · FREE

설치부터 나만의 워크플로우까지 —
매일 쓰는 사람이 정리한 10편을, 카드보다 한 단계 깊게 PDF 한 권으로.

무료 배포 · 공식 문서(2026-06) 기준 검증 · @sun.young.0207

@sun.young.0207

CONTENTS

10편에 담긴 것

각 편은 IG·Threads 카드의 '무엇'에 더해, 카드엔 못 담은 **왜 · 언제 · 판단 기준 · 한 걸음 더**를 한 쪽에 담았어요.
기술 사실은 공식 문서(code.claude.com/docs) 기준으로 검증했어요.

01

첫걸음

Claude Code 첫 5분

설치는 **30초**면 끝나요. 진짜 시작은 그다음 — "이 친구한테 일을 어떻게 맡길지" 감을 잡는 **5분**이에요. **Claude Code는 매 세션 처음부터 함께 일하는 협업 상대**라, 첫날 잡은 **CLAUDE.md**(규칙)와 **권한 감각**(어디까지 맡길지)이 1년을 좌우해요.

첫날 챙길 3가지 — 이거면 충분해요

- 설치 → `cd 프로젝트` → `claude`. 첫 명령은 `/init` — 프로젝트를 분석해 **CLAUDE.md 초안**을 자동 생성해요
- 권한은 `default`에서 시작 — `Shift+Tab`으로 전환. 손에 익기 전엔 한 단계씩 확인이 안전해요
- `/help` — 명령어는 외우지 말고 필요할 때 꺼내 써요

언제 뭘 쓰나 — 권한 모드 판단 기준

새 코드를 처음 파악	→	`plan` — 읽기 전용으로 먼저 훑기
변경을 막 시작	→	`default` — 한 단계씩 확인하며 익히기
흐름이 손에 익으면	→	`acceptEdits` — 그때 속도 풀기

왜 순서가 핵심이에요. **익숙해진 뒤 → 빠르게**지, 첫날부터 자동은 어디서 틀어졌는지 못 짚어요.

한 걸음 더 — 첫 5분 습관의 정체

명령어 암기가 아니에요. **"내가 기대하는 결과를 먼저 한 줄로 말하고 시키는 것"** — 이 습관 하나가 초보와 능숙함을 가르고, 나머지 9편이 전부 이 위에 쌓여요.

@sun.young.0207

02

규칙

CLAUDE.md 실수 5가지

CLAUDE.md는 매 세션 자동 로드되는 **프로젝트 헌법**이에요. 그런데 있어도 효과가 없는 데는 이유가 있어요 — 대부분 "쓰는 법"이 아니라 **"쓰는 태도"** 문제예요.

효과 못 내는 5가지 이유 → 해결

- ① 빈 파일만 만듦 → **1줄이라도 ("응답은 한국어로")**
- ② 완벽하게 쓰려다 포기 → **1줄로 시작, 발견할 때마다 추가**
- ③ 바뀌어도 안 고침(낡음) → **커밋할 때 같이 갱신**
- ④ "잘 해줘" 같은 소원형 → **행동 단위로 ("편집 전 파일 읽기")**
- ⑤ 이유 없는 명령 → **"— 이유: ..." 한 줄 덧붙이기**

소원형 ❌ vs 규칙형 ✅ — 왜 갈리나

잘 해줘 · 꼼꼼히 · 정확하게	✕ 행동으로 못 옮겨요 (해석 여지가 큼)
응답 한국어로 · 테스트 없이 배포 금지	✓ 검증 가능한 행동이라 그대로 지켜져요

왜 Claude는 문장을 **문자 그대로** 해석해요. 모호한 소원은 빈칸으로 남고, 행동 지시는 그대로 실행돼요.

한 걸음 더 — 살아있는 문서

완벽한 CLAUDE.md는 없고 **살아있는 CLAUDE.md**만 있어요. 길이가 아니라 **"이 줄 빼면 실수하나?"**로 매 줄을 검문하세요. 길수록 매 세션 비용이 올라가요.

@sun.young.0207

03

컨텍스트

Context Window 관리법

Claude의 **단기 작업 공간**이에요 (표준 **200K 토큰**, 일부 모델은 1M 선택 가능). 꽉 차면 오래된 것부터 잊기 시작해요. **맥락 관리는 Claude가 아니라 내 일**이에요.

공간을 채우는 것 + 4가지 명령

- 대화 내역 + 읽은 파일·코드 + CLAUDE.md·MCP가 전부 쌓여요
- `/context` 무엇이 차지하나 분석 · `/usage` 사용량·비용
- `/compact` 요약 압축 (**맥락 유지**, 오래된 출력만 제거) · `/clear` 완전 초기화

이 신호 오면 → 이 명령

답이 느려짐·잘림 / 앞 얘길 모름 / 한 일을 또 함	→	`/compact`
전혀 다른 새 주제로 전환	→	`/clear`

왜 같은 작업이면 **절대 /clear 금지** — 맥락까지 버려요. /compact는 맥락은 두고 오래된 출력만 덜어내요.

한 걸음 더 — 많이 ≠ 잘

토큰이 늘수록 오히려 **recall이 떨어져요** (context rot). **"더 많이 넣으면 더 잘하겠지"는 함정** — 필요한 것만 좁게 주는 게 더 똑똑한 사용법이에요.

@sun.young.0207

04

권한

툴 허용 — 6가지 권한 모드

Claude가 파일·셸을 **얼마나 자동으로** 다룰지 정하는 거예요. `Shift+Tab`으로 즉시 전환, `settings.json`으로 영구 설정. **속도와 안전의 다이얼**이라고 보면 돼요.

6가지 모드

- `default` 변경 전 확인 (첫날 권장) · `acceptEdits` 파일 편집 자동 수락
- `plan` 읽기 전용, 계획만 · `auto` 자동 + 백그라운드 안전 체크
- `dontAsk` 사전 승인(allow) 외 자동 거부 · `bypassPermissions` 모든 확인 생략 **(rm -rf /·~만 안전장치)**

SETTINGS.JSON — 영구 규칙 + 왜 DENY 우선

- `permissions`의 `allow` / `ask` / `deny` 세 배열로 세분화
- `Bash(npm run build)`처럼 **Tool(specifier)** 문법 + 와일드카드 `*`
- 프로젝트 `.claude/` · 개인 `~/.claude/`, **deny가 항상 우선** — 실수로 allow를 넓혀도 deny가 안전망이 돼요

한 걸음 더 — 권한은 신뢰 곡선

첫날 **default** → 팀 프로젝트는 **deny부터**(공유 위험 차단) → 개인 반복은 `~/.claude/`의 allow. **한 번에 bypass로 점프하는 게 가장 큰 사고**예요.

@sun.young.0207

05

멀티파일

코드 전체를 Claude 눈에

Claude는 **"멀티파일 모드"가 따로 없어요.** 프롬프트에 경로·패턴을 명시하는 게 전부예요. 즉 **무엇을 보여줄지 정하는 건 나**예요.

실전 패턴 4가지

- 파일 경로 직접 — `app/api/users/route.ts 수정해줘`
- Glob 패턴 — `src/**/*.ts 에서 TODO 찾아줘`
- 큰 작업은 `plan` 모드로 먼저 탐색 → 합의 후 편집
- 한 번에 10개 ❌ → **논리 단위로** 나눠 요청

안전한 멀티파일 루틴 — 순서가 곧 안전망

① plan 탐색	→	건드릴 파일·영향 범위 먼저 파악
② 기대 결과 선언	→	"이렇게 되면 성공" (테스트면 더 확실)
③ 편집 → 되돌리기	→	`Esc` 두 번 — 체크포인트로 복원

왜 기대 결과를 먼저 말하면 Claude가 **목표를 알고** 편집해요. 헤매면 모델이 멍청한 게 아니라 안 보여준 거예요.

한 걸음 더 — CLAUDE는 본 것만 안다

막힐 때 첫 디버깅은 프롬프트 수정이 아니라 **"경로·패턴을 더 좁혀서 정확히 보여줬나"** 점검이에요. 시야를 정해주는 게 절반이에요.

@sun.young.0207

06

연결

MCP 연결 처음 해보기

MCP(Model Context Protocol)로 외부 도구·서비스를 Claude에 연결해요. GitHub 이슈 조회·PR 생성까지 Claude에서 직접. 단, **연결만 해도 컨텍스트를 먹어요.**

4가지 종류 — 어디서 왔나

- **Project** — `.mcp.json` (프로젝트 루트, git 공유)
- **User** — `~/.claude.json` (내 모든 프로젝트)
- **claude.ai** — 웹에서 추가한 통합 (Canva·Notion 등)
- **Built-in** — 내장 (예: computer-use, 기본 비활성)

실전 3규칙 + 왜

- 안 쓰는 MCP는 끄기 — 연결만 해도 매 세션 컨텍스트를 차지해요
- `npx` 붙은 서버부터 — Node.js만 있으면 설치 없이 실행돼요
- `/mcp`의 **△ needs authentication** = 코드 문제 아님, **토큰 발급이 먼저**

한 걸음 더 — 서버 수 = 고정비용

서버 하나하나가 매 세션 깔리는 **고정 오버헤드**예요. **"있으면 좋겠지"로 늘리면 느려져요** — 이번 작업에 진짜 쓰는 **1~2개**만 켜고 시작해요.

@sun.young.0207

07

기억

memory 시스템 — 3층 메모리

Claude Code의 메모리는 **3개 독립 레이어**예요. 각자 다른 생명주기로 로드돼요 — **어디에 적느냐가 효과를 갈라요.**

3층 구조

- **L1 CLAUDE.md** — 내가 씀. 세션 시작 시 전체 자동 로드 (`./CLAUDE.md` · `~/.claude/CLAUDE.md`)
- **L2 Auto memory** — Claude가 씀. `MEMORY.md` **앞 200줄·25KB** 자동 로드
- **L3 .claude/rules/*.md** — 경로가 매칭될 때만 로드되는 특화 규칙

/MEMORY + 실전 3규칙

- `/memory` — 로드된 메모리 확인 + 자동 메모리 on/off
- CLAUDE.md는 **200줄 이하** — 길수록 매 세션 비용↑·recall↓, 상세는 `@파일명` 으로
- Auto memory는 Claude의 것 — 직접 편집 말기 / **민감정보 금지 (평문 저장)**

한 걸음 더 — 적립보다 '제때 로드'

메모리의 핵심은 많이 쌓는 게 아니라 **필요할 때 읽히는 것**이에요. **항상 필요 → L1, 경로별 → L3, Claude 자율 → L2.** 자리를 틀리면 아무리 적어도 안 읽혀요.

@sun.young.0207

08

분신

Sub-agent 파견하기

메인은 **조율자**, 일은 분신이. 분신은 자기만의 컨텍스트에서 처리하고 **요약만** 돌려줘요. 그래서 **메인 맥락이 오래 또렷하게** 유지돼요.

왜 쓰나 + 만들기

- 왜 — ① 병렬(독립된 일 동시에 = 속도) ② 격리(메인엔 요약만 = 절감)
- 만들기 — `/agents` 대화형, 또는 `.claude/agents/이름.md` 파일
- 필수는 `name` · `description` . `tools` 는 **콤마 문자열** (`Read, Grep` — 배열 아님)

핵심 3규칙 (함정 주의)

- `/agents` 또는 파일로 만들기 — **/agents create 같은 명령은 없어요**
- 분신 안에서 또 분신 금지 (중첩 X — `tools` 에 `Agent` 넣어도 무효)
- 원본 말고 **요약만** 전달 — 격리가 곧 절감이에요

한 걸음 더 — 판단 한 문장

"이 일, 메인 맥락 없이 요약만으로 되나?" 예 → 분신. 진짜 가치는 속도보다 **메인 컨텍스트 보호**예요. 단, 짧은 일은 오히려 오버헤드 손해라 출력이 길거나 병렬일 때만.

@sun.young.0207

09

비용

비용 최적화 전략

비용은 **모델 선택에서 80%** 결정돼요. 먼저 현황을 보고, 작업에 모델을 맞춰요. **"모르니까 Opus"가 가장 큰 낭비**예요.

모니터 2종 + 모델 라우팅

- `/usage` 세션 비용·한도·무엇이 갉아먹나 · `/model` 전환 + effort 조절
- `haiku` — 파일 탐색·검색·로그 (이해 필요 없는 일)
- `sonnet` — 코딩·리팩토링·문서 (**기본값, 대부분 충분**)
- `opus` — 아키텍처·복잡한 판단 (아낄 것, 결정이 필요할 때만)

절약 전략 + 왜 효과가 크나

- `/compact` 자주 · MCP 정리 · 구체적 프롬프트 · plan 모드 먼저
- 대용량은 Sub-agent로 격리 · 무관한 작업 쌓이면 `/clear`
- 비용 = 토큰 × 모델 단가. Opus는 sonnet의 수 배라, 판단 아닌 일에 쓰면 그대로 새요

한 걸음 더 — 충분한 것을 고른다

모델은 "가장 똑똑한 것"이 아니라 **"이 작업에 충분한 것"**을 골라요. Sonnet 기본, 탐색 Haiku, 판단할 때만 Opus — 이 한 줄이 월 비용을 가장 크게 바꿔요.

@sun.young.0207

10

자동화

나만의 워크플로우 만들기

커스터마이징은 **3층**으로 쌓여요. 아래에서 위로 갈수록 더 복잡한 걸 자동화해요. **처음부터 다 만들 필요는 없어요.**

3층 구조

- **L1 CLAUDE.md** — 매 세션 자동 로드되는 규칙 (`@import` 로 분리, `.claude/rules/` 로 경로별)
- **L2 Hooks** — 생명주기 이벤트에 셸 명령 자동 실행 (`settings.json`)
- **L3 Commands·Skills** — `/이름` 으로 부르는 나만의 단축 명령

만들기 — 작게 시작

- 지시문 한 장이면 `.claude/commands/이름.md`
- 부속 파일·자동 호출이 필요하면 `.claude/skills/이름/SKILL.md` 로 승격
- `!`명령`` 문법으로 셸 결과를 프롬프트에 자동 주입

한 걸음 더 — 반복이 보일 때 한 층씩

처음부터 3층을 다 설계하면 **안 쓰는 추상화**만 쌓여요. **가장 자주 하는 것 하나를 명령으로** 만드는 데서 시작 — 거기서 필요할 때마다 한 층씩 얹어요.

@sun.young.0207

KAIROS AI STUDIO · CC-TIPS

여기까지 보셨다면
이미 잘 쓰고 계신 거예요

설치부터 나만의 워크플로우까지 10편 —
이제 매일 쓰면서 **손에 익히는 일**만 남았어요.

매주 새로운 Claude Code 팁을 카드로 발행해요

Instagram · Threads **@sun.young.0207**

홈페이지 **kairos.jeju.at**