


Claude Code

처음 시작할 때 7가지 설정



Index

목차소개

- 01 언어 설정 - 한국어
- 02 터미널 표시 설정 - 작업 상황을 항상 파악
- 03 개인정보 보호 설정 - 모델 학습 미사용 확인
- 04 CLAUDE.md 작성 - 프로젝트를 이해 향상
- 05 권한 설정 - 승인 피로감 감소
- 06 훅 설정 - 고급 자동화

김재우

jaewoo@mdrules.dev

01 언어 설정 - 한국어

무엇이 바뀌는가?

설정하면 Claude Code의 응답이 전부 한국어가 된다. 코드내의 주석제안, 에러메시지 설명도 한국어로 나오기 때문에, 영어에 익숙하지 않아도 스트레스 없이 사용할 수 있다.

설정방법

- 1_Claude Code를 실행 후, /config 명령을 실행한다.
- 2_'Language' 항목으로 이동후, 스페이스바를 눌러 선택한다.
- 3_"Korean"으로 입력하고 변경완료!

02 지금 상태를 확인하기 - 상태 라인

터미널 표시 설정 (상태 라인)

구분	내용
모델명	사용중인 모델 (Opus, Sonnet 등)
작업 디렉토리	현재 경로와 Git 브랜치
컨텍스트 크기	토큰 소비량과 사용률
비용	세션 추정 비용
속도 제한	5시간/7일 사용률
버전	Claude Code 버전 번호

설정 방법

- 1_ "상태라인을 설정해" 또는 "statusline에 무엇이 표시할 수 있는지 알려줘"라고 입력해 보자.
- 2_ 설정스크립트를 자동 생성해준다.

03 데이터 재학습 여부 확인 - 개인정보 설정

무엇이 바뀌는가?

Claude Code는 기본값으로 모델 재학습에 데이터 사용하도록 설정되어 있다. 단, Team/Enterprise 계정인 경우, 기본값이 사용하지 않도록 설정되어 있다. 그래도 업무내용이나 프라이빗한 내용을 취급하는 경우, 만약을 위해 확인하고자 할 때 사용한다.

설정방법

- 1_Claude Code를 실행 후, /privacy-setting 명령을 실행한다.
- 2_”Data Privacy”내용에 “Help improve Claude”가 “false”로 되어 있다면 안심하고 넘어가자.
- 3_안되어 있다면 Space바를 눌러 값을 false로 변경하자.

04 매번 같은 말을 하지 않도록 - CLAUDE.md

무엇이 바뀌는가?

각 세션이 새로운 컨텍스트 윈도우로 시작된다. CLAUDE.md 파일을 배치하는 것만으로 프로젝트의 구조, 코딩규약, 빌드명령등의 영속적인 지시를 매회 자동적으로 Claude Code에 전할 수 있어, 매 세션에서 같은 설명을 반복할 필요가 없어진다.

/init명령은 사용하지 않고 직접 작성을 권장한다. 이 명령을 실행하면 코드베이스를 분석하여 CLAUDE.md를 자동생성하지만 자동생성된 내용이 너무 범용적이고 실용성이 낮은 케이스가 많아 비추천한다.

WHAT/WHY/HOW기준으로 구성하자.

추천템플릿

프로젝트 이름

Next.js 16 전자상거래 앱. App Router, Turbopack, Stripe 결제 사용.

명령

- `pnpm dev`: 개발 서버
- `pnpm test`: Vitest 실행
- `pnpm test : e2e`: Playwright 실행

코드 스타일

- TypeScript strict 모드
- named export 사용

아키텍처

- `/app`: 페이지와 레이아웃
- `/app/api`: API 루트 (`/api/v1` 접두사)
- `/components/ui`: UI 구성 요소

주의사항

- Stripe webhook은 서명 검증 필수
- 자세한 내용은 @docs/auth-flow.md를 참조하십시오.

05 같은 승인 요청을 줄이자 - 권한 설정

무엇이 바뀌는가?

Claude Code는 기본적으로 파일 편집이나 명령 실행할 때마다 허가를 요구한다. 반복적인 승인 조작이 생산성을 저하시킬 수 있다.

allow/deny/ask의 3계층 제어

settings.json의 permissions개체로 3계층 제어를 한다. 평가순서는 deny → ask → allow로 deny가 항상 최우선이다.

추천 사용권한 (전역: ~/.claude/settings.json)

```
{
  "permissions": {
    "allow": [
      "Bash(git log *)",
      "Bash(git diff *)",
      "Bash(git status *)",
      "Bash(git branch *)",
      "Bash(* --version)",
      "Bash(* --help *)"
    ],
    "deny": [
      "Bash(curl *)",
      "Bash(wget *)",
      "Bash(rm -rf *)",
      "Bash(sudo *)",
      "Bash(chmod 777 *)",
      "Read(/.env)",
      "Read(/.env.*)",
      "Read(/secrets/**)"
    ]
  }
}
```

05 같은 승인 요청을 줄이자 - 권한 설정

프로젝트 설정: .claude/settings.json

```

{
  "permissions": {
    "allow": [
      "Bash(npm run lint)",
      "Bash(npm run test *)",
      "Bash(npm run build)",
      "Bash(npx prettier *)"
    ],
    "deny": [
      "Bash(git push *)",
      "Bash(npm publish *)"
    ]
  }
}

```

Shift + Tab키로 모드 전환

모드	내용	추천 장면
default	매번 허가 요청	초기 설정, 신중한 작업
acceptEdits	파일 편집은 자동 승인	코드 수정의 반복
plan	읽기 전용. 변경하지 않음	코드베이스 탐색, 설계 검토
auto	분류기 모델이 안전성을 판단하고 자동 승인	장시간 작업

초보자는 우선 default 조작에 익숙해지고, 익숙해지면 acceptEdits이행하는데 그래. "Yes, don't ask again"을 선택하면 자주 사용하는 명령이 자동으로 allow 목록에 추가됩니다.

05 같은 승인 요청을 줄이자 - 권한 설정

자동 모드 - dangerously-skip-permissions의 안전한 대안 (2026년 3월 24일 추가)

비교항목	auto 모드	dangerously-skip-permissions
안전 점검	분류기는 백그라운드에서 평가	없음. 모두 즉시 실행
프롬프트 인젝션 대책	예	없음
에스컬레이션	3회 연속 블록으로 인간에게 위양	없음
계획 요건	Team 플랜 이상	없음(모든 플랜)
모델 요구 사항	Sonnet 4.6/Opus 4.6 전용	모든 모델

auto 모드 분류기는 2단계로 동작한다.

1_State1(고속필터): 단일토큰으로 yes/no판정, 고속이미지가 블록에 가깝게 조정된다.

2_State2(사고연쇄추론): State1에서 플래그된 것만 상세 평가, 거짓 긍정을 0.4%로 줄인다.

분류기는 Claude의 추론 텍스트와 툴 결과를 의도적으로 보이지 않게 설계된다.

보상상 권한 설정은 Claude Code의 의사결정을 제어하는 소프트웨어 계층이며 OS레벨의 강제등으로 'Read(./env)' deny룰은 Read툴을 블록하는 것이지만 'Bash(cat .env)와 같은 것은 블록을 막을 수 없다.이런 경우를 대비해 샌드박스과 같이 이용하는 것을 추천한다.

06 어떤 모델을 사용할까? - 성능 설정

모델 선택

선택	모델	API요금 (I/O)	컨텍스트	특징
Default	Opus 4.6	\$5/\$25	1M	최고의 추론력
Sonnet	Sonnet 4.6	\$3/\$15	200K	일상 업무에 이상적
Sonnet(1M)	Sonnet 4.6	\$3/\$15	1M	대규모 코드베이스 방향
Haiku	Haiku 4.5	\$1/\$5	200K	가장 빠른 · 최저

전환 방법

- 1_ '/model' 명령을 실행한다.
- 2_ 시작할 때 'claude --model' 옵션 또는 /config의 model필드에서 변경할 수 있다.

06 어떤 모델을 사용할까? - 성능 설정

Effort Level - 생각의 깊이 제어

레벨	권장작업	토큰소비
low	파일 이름 바꾸기, 변수 이름 바꾸기, 구문 확인	최소
medium (기본값)	일상 코딩, 함수 생성, 테스트 생성	보통
high	복잡한 디버깅, 다중 파일 리팩토링	많은
max (Opus 4.6전용)	가장 어려운 분석, 100+ 파일의 대규모 변경	최대

사용법

Claude Code가 각 응답에 얼마나 깊게 생각하는지를 제어하는 파라미터설정이다.

/effort low ~ /effort max로 변경할 수 있다.

태스크의 복잡성에 따라 구분하는 것만으로 전부 high로 실행하는 경우와 비교해 비용을 약 40~60% 절약할 수 있다.

06 어떤 모델을 사용할까? - 성능 설정

고속 모드 - 빠른 추론

모드	입력(MTok)	출력(MTok)	속도
Standard Opus 4.6	\$5	\$25	표준
Fast Mode Opus 4.6	\$30	\$150	2.5배

출력 토큰 비용이 약 6배부터, 대화적 디버그나 마감 작업 같은 시간적 가치가 코스트를 웃도는 장면 에서 사용하기 시작했다. 세션 도중에 활성화하면 대화 컨텍스트가 fast mode 가격으로 재계산 되어 버리기 때문에, 세션 개시시에 유효하게 하는 것이 제일 비용 효율이 좋다. 도중에 하면 손해이다.

사용법

/fast 명령으로 Opus 4.6의 고속 추론 모드를 활성화할 수 있다.

같은 모델품질로 최대 2.5배속의 응답을 얻을 수 있다.

07 실행하기 전에 멈춘다 - 후 설정

후 핸들러

유형	설명
command	셸 명령 실행 (가장 자주 사용하는 사람)
http	HTTP 엔드포인트로 POST
prompt	LLM에 의한 단일 턴 판정
agent	멀티턴 검증(서브에이전트 시작)

금지명령 블록 - PreToolUse로 특정 커맨드 실행막음

```
{
  "hooks": {
    "PreToolUse": [{
      "matcher": "Bash",
      "hooks": [{
        "type": "command",
        "command": "if echo \"${CC_BASH_COMMAND}\" | grep -qE 'rm -
rf|sudo|curl'; then exit 2; fi"
      }]
    }]
  }
}
```