AI 바이브/페어 개발 — CLAUDE CODE/CODEX 과정

Claude Code 셋업 명령어·모드·CLI 완전정복

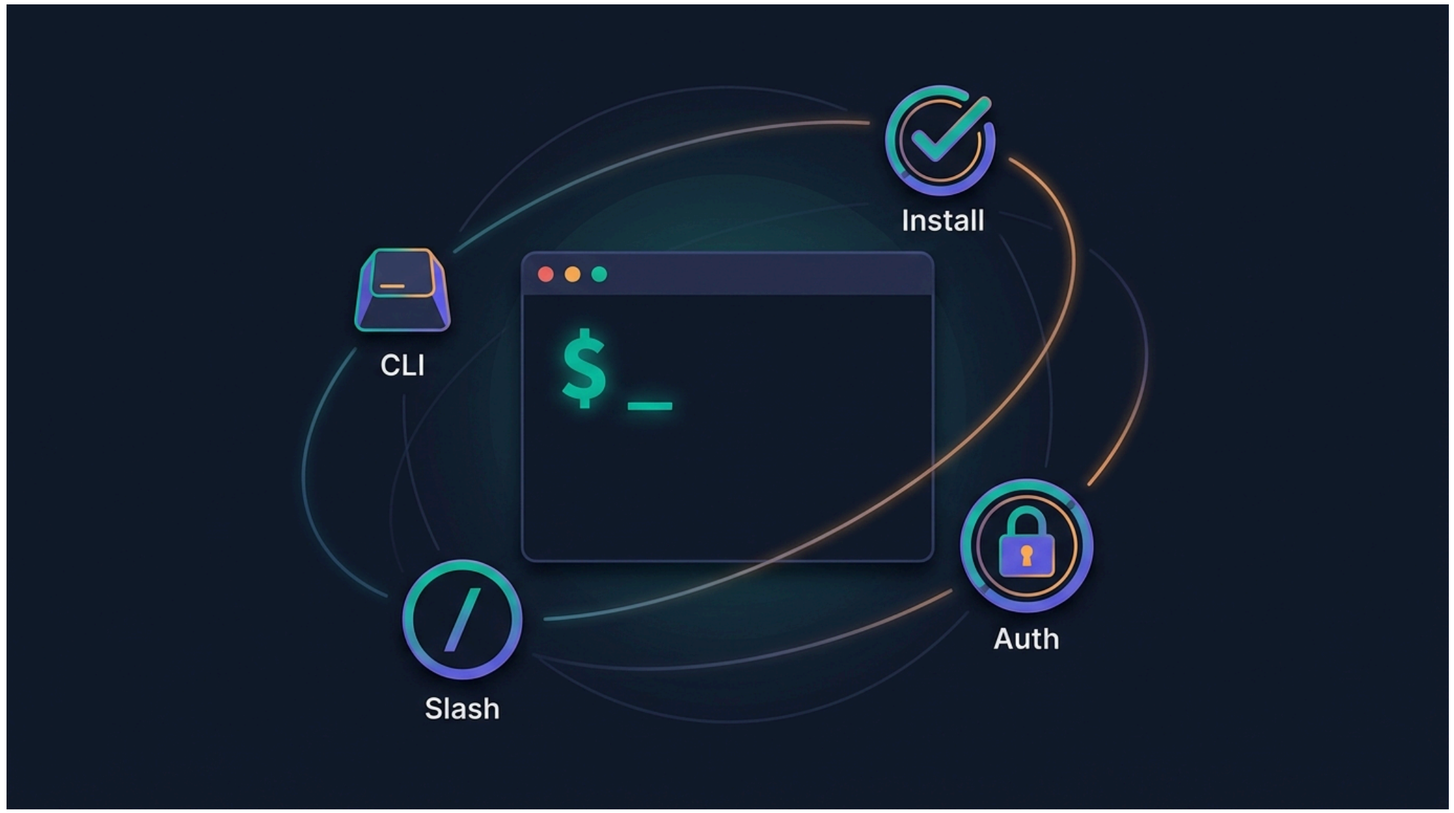
Session 1 · Claude Code 셋업·명령어·모드·CLI

설치·인증 · 모드 5종 · 슬래시 20+ · CLI 인자 · @file·!cmd · 조합 매트릭스

강사 박수현 · 🚀 젠아이랩스(GenAI Labs)

바이브코딩 2일 집중 · 프롬프트·컨텍스트·하네스 엔지니어링

출처: Claude Code 공식 문서 Anthropic Docs



Session 1 · Claude Code 셋업·명령어·모드·CLI

© 2026 젠아이랩스 (GenAI Labs) · 강사 박수현 · v0.1.0 · c2b23b7

PART 1

셋업 — *설치·인증·모델 선택*

> 설치 경로 3종 · 플랫폼별 유의점 · 첫 인증 · `~/.claude` 구조 · 구독·요금 · 모델 3종 · 첫 대화 5분.

출처: Claude Code 공식 문서 | Anthropic Docs

설치 경로 3종 — *무엇을 언제 쓰나*

Native installer (권장)

```
curl -fsSL https://claude.ai/install.sh | bash
```

- **자동 업데이트** — 예 (백그라운드)
- **언제** — 대부분의 상황 · macOS · Linux · WSL

Homebrew

```
brew install --cask claude-code
```

- **자동 업데이트** — 아니오 (수동)
- **언제** — macOS에서 brew를 이미 쓰고 있을 때

npm

```
npm install -g @anthropic-ai/claude-code
```

- **자동 업데이트** — 예 (백그라운드)
- **언제** — Node 18+가 있고 툴을 통합 관리하고 싶을 때

📌 **주의 세 가지.** - **Native installer는 2026년 3월 이후 공식 권장.** 의존성 없이 macOS · Linux · Windows 모두 지원한다. - `sudo npm install -g` **는 금지.** 권한 문제로 이후 업데이트가 깨진다. nvm을 쓰거나 native installer로 간다. - **Homebrew는 자동 업데이트가 없다.** `brew upgrade claude-code`를 주기적으로 돌린다.

© 2026 젠아이랩스 (GenAI Labs) · 강사 박수현 · v0.1.0 · c2b23b7

출처: Claude Code 공식 문서 | Anthropic Docs

플랫폼별 유의점 — *macOS · Linux · WSL2 · Windows*

플랫폼	시스템 요구	설치 팁	샌드박스
macOS 13+	ARM · x64 모두	Native installer 또는 Homebrew	지원
Ubuntu 20+ · Debian 10+	Bash · Zsh · 4GB+ RAM	Native installer 또는 apt/dnf/apk	지원
WSL 2	WSL 2 활성화	WSL 터미널 안에서 Linux installer	지원
Windows Native	Windows 10 1809+ · PowerShell/CMD	PowerShell installer (아래 참고)	**미지원**

Windows Native — PowerShell

```
irm https://claude.ai/install.ps1 | iex
```

🪟 Windows 사용자의 선택

- **WSL 2** — Linux 툴체인·샌드박스가 필요할 때. **이 강의 권장.**
- **Windows Native** — Windows 전용 프로젝트. Git for Windows가 있으면 Bash tool을 쓴다. 샌드박스는 지원하지 않는다.

🐧 Alpine · musl 배포판

- `libgcc` · `libstdc++` · `ripgrep`을 별도로 설치한다
- `USE_BUILTIN_RIPGREP=0` 환경변수가 필수다

© 2026 젠아이랩스 (GenAI Labs) · 강사 박수현 · v0.1.0 · c2b23b7

출처: Claude Code 공식 문서 | Anthropic Docs

데스크톱 앱 최근 변경 — *Home 통합 · Cowork · Hyper-V*

> **Chat·Cowork가 Home 탭 하나로 통합**됐다. Cowork는 가상화 위에서 돌아 Windows는 별도 준비가 필요하다.

🏠 Home 탭 통합 UI

- 분리돼 있던 **Chat · Cowork** 탭이 **Home 탭 하나**로 통합됐다
- **입력창 하단 토글**로 두 모드를 전환한다
- 같은 대화 흐름에서 오가므로 컨텍스트 유지가 편하다

☁️ 원격 세션 (베타)

- Cowork 세션이 **Anthropic 서버에서 실행**된다
- **노트북을 닫아도 지속**되고, **데스크톱 · 웹 · 모바일** 어디서 열어도 같은 세션에 연결된다
- 오래 걸리는 리팩토링을 걸어두고 이동해도 이어서 확인한다

📌 **로컬 리소스가 필요한 작업은 여전히 데스크톱 앱이 필수. 로컬 폴더** 코드 작업 · **로컬 MCP** 서버(파일시스템·Docker 등) · **브라우저 자동화·컴퓨터 사용(Computer Use)** 은 원격 세션으로 처리되지 않는다.

🪟 Windows Hyper-V 요구사항 — Cowork를 켤 수 있는가

Cowork는 가상화 위에서 격리 실행된다. Windows에서 쓰려면 **Hyper-V 또는 Windows 하이퍼바이저 플랫폼**이 켜져 있어야 한다.

- **Windows 10 Home · 저가 SKU** — Hyper-V 옵션이 없다 → **WSL 2 설치로 우회**(하이퍼바이저 플랫폼이 함께 켜진다)
- **Windows 11 Pro · Enterprise** — 「Windows 기능」에서 Hyper-V · 가상 머신 플랫폼 · 하이퍼바이저 플랫폼 체크 후 재부팅
- **CPU 가상화(VT-x · AMD-V)** 는 BIOS/UEFI에서 켜져 있어야 한다

© 2026 젠아이랩스 (GenAI Labs) · 강사 박수현 · v0.1.0 · c2b23b7

출처: Claude Code 공식 문서 | Anthropic Docs

설치 확인 · 첫 인증

```
# 1. 설치 확인
claude --version

# 2. 환경 진단 (권장)
claude doctor

# 3. 첫 실행 · 인증 (브라우저가 열림)
claude
```

🔐 인증 요건 — 세 옵션

1. **Pro / Max 구독** — Claude.ai 개인 계정 로그인
2. **Team / Enterprise 조직** — 조직 계정 로그인
3. **Console (API 종량 발급 포털)** — `claude auth login --console`

⚠️ **무료 Claude.ai 플랜은 사용 불가.**

✅ 인증 성공 확인

- 프롬프트가 `>`로 바뀐다
- 우상단에 현재 모델명이 뜬다 (`claude-sonnet-4-6` 등)
- 좌하단에 현재 디렉토리·git 브랜치가 뜬다

📌 **인증 실패 상위 3가지.** ① 프록시·방화벽에서 브라우저 리다이렉트 실패 → 개인 hotspot으로 우회. ② 이전 계정 잔여 세션 → `claude auth logout` 후 재시도. ③ 지원 국가 밖 접속 → VPN 정책 확인.

© 2026 젠아이랩스 (GenAI Labs) · 강사 박수현 · v0.1.0 · c2b23b7

출처: Claude Code 공식 문서 | Anthropic Docs

`~/.claude` 디렉토리 구조

```
~/.claude/
├── CLAUDE.md          # 전 프로젝트 공통 개인 지침 (선택)
├── settings.json      # 설정 · 권한 · 훅
├── projects/          # 프로젝트별 세션 로그
│   └── -home-user-repo/
├── skills/            # 개인 스킬 정의
├── plugins/           # 설치한 플러그인
├── statsig/           # 원격 설정 캐시
└── ...
```

🧠 컨텍스트 축 진입 파일

`~/.claude/CLAUDE.md`

모든 프로젝트에서 공유하는 개인 지침. 컨텍스트 엔지니어링에서 가장 먼저 만지는 파일이다.

🔧 하네스 축 진입 파일

`~/.claude/settings.json`

자동 업데이트 채널·권한·훅. 하네스 엔지니어링에서 가장 먼저 만지는 파일이다.

📌 **두 축은 이 슬라이드에서 파고들지 않는다. 컨텍스트 축(CLAUDE.md·스킬·서브에이전트)** 과 **하네스 축(권한·훅·MCP)** 은 이 코스의 별도 세션에서 자세히 다룬다. 여기서는 **진입 파일 위치만 익혀 둔다.**

⚠️ **주의.** `rm -rf ~/.claude`를 하면 세션 이력·설정·허용 툴·MCP 설정이 모두 지워진다. 재설치가 목적이면 백업이 필수다.

© 2026 젠아이랩스 (GenAI Labs) · 강사 박수현 · v0.1.0 · c2b23b7

출처: Claude Code 공식 문서 | Anthropic Docs

구독·요금 — *2026년 6월 기준*

플랜	월 요금	Pro 대비 사용량	대상
Pro	$20	기준	개인 · 실험 · 이 강의
Max 5x	$100	5배	매일 쓰는 개발자 · 리팩토링 잦은 팀
Max 20x	$200	20배	대형 리포 · 장시간 세션 · Subagent 병렬
API (종량)	별도	무제한 (비용대로)	CI · 자동화 · 팀 전체 배포

📌 **주의 세 가지.** - 요금·한도는 자주 바뀐다. **2026년 6월 기준**임을 명시한다. - Max는 **주간 사용량 한도**도 있다. 전 모델 통합 한도와 Sonnet 전용 한도가 별도로 걸린다. - API 종량은 Claude 구독과 별개 청구. `claude auth login --console`로 API 인증 후 쓴다.

© 2026 젠아이랩스 (GenAI Labs) · 강사 박수현 · v0.1.0 · c2b23b7

출처: Claude Code 공식 문서 | Anthropic Docs

모델별 API 요금 — *Anthropic 공개, per MTok*

모델	입력 (per MTok)	출력 (per MTok)
Opus 4.7	$5	$25
Sonnet 4.6	$3	$15
Haiku 4.5	$1	$5

💰 Opus

5배 비싸다

Sonnet 대비 입력·출력 모두 약 5배. 실패 비용이 큰 결정에만 쓴다.

⚖️ Sonnet

기본값

성능·비용 균형점. 일상 개발의 표준.

🚀 Haiku

5배 저렴하다

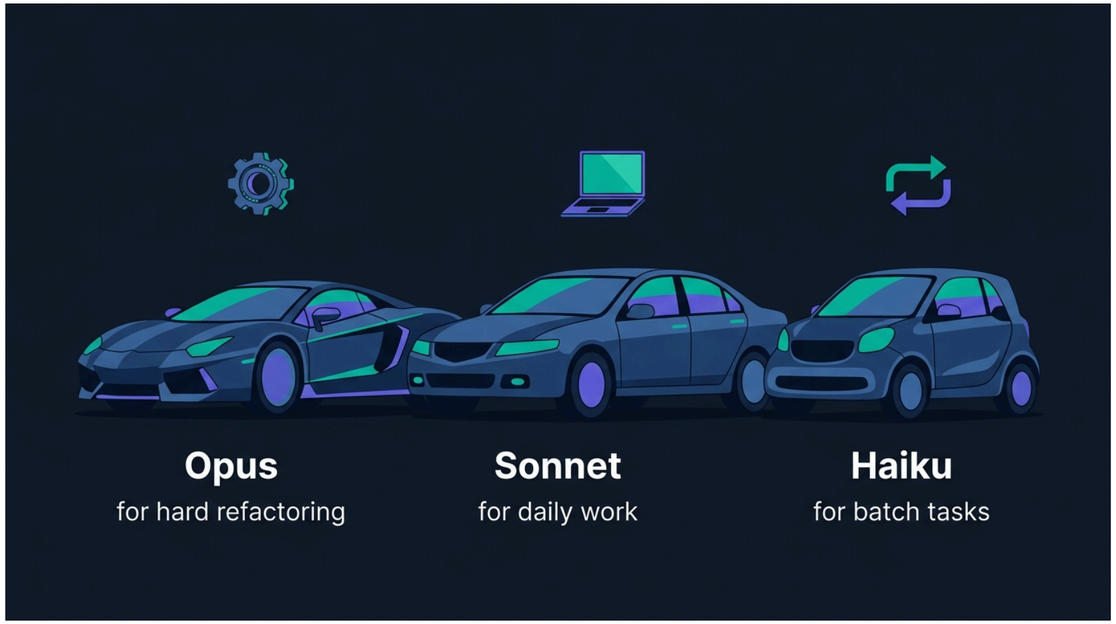
Sonnet 대비 입력·출력 모두 약 1/5. 반복·CI 배치에 쓴다.

© 2026 젠아이랩스 (GenAI Labs) · 강사 박수현 · v0.1.0 · c2b23b7

출처: Claude Code 공식 문서 | Anthropic Docs

모델 3종 실무 감각 — *Opus · Sonnet · Haiku*

모델	강점	실무 태스크	SWE-bench
Opus 4.7	최고 추론 · 에이전트 자율성	어려운 리팩토링 · 아키텍처 결정 · 대형 마이그레이션	~80%
Sonnet 4.6	성능·비용 균형	일반 개발 · 코드 리뷰 · 기본값	~79.6%
Haiku 4.5	저비용 · 초고속	반복 · 린트 · CI · 대량 검색	~73.3%



📌 **태스크별 라우팅 원칙.** - **탐색·설계·리팩토링** → Opus. 실패 비용이 큰 결정에 쓴다. - **일상 개발·수정·리뷰** → Sonnet. 기본값. - **반복·자동화·저비용 배치** → Haiku. CI 파이프라인·대량 grep을 대체한다. - **비용이 걱정되면** → `/cost`로 확인하고 모델을 낮춘다.

© 2026 젠아이랩스 (GenAI Labs) · 강사 박수현 · v0.1.0 · c2b23b7

출처: Claude Code 공식 문서 | Anthropic Docs

모델 전환 두 가지 · Fallback chain

```
# 1. 세션 시작 시
claude --model sonnet          # 별칭: opus · sonnet · haiku
claude --model claude-sonnet-4-6   # 정식 모델 ID

# 2. 세션 안에서
/model sonnet

# 3. Fallback — 과부하 시 자동 전환
claude --fallback-model sonnet,haiku
```

🔀 세션 안 전환

`/model` 슬래시 명령으로 즉시 바꾼다. 어려운 결정 지점에서 잠깐 Opus로 올렸다가, 반복 작업으로 넘어가면 다시 Sonnet으로 내린다.

🛟 Fallback chain

특정 모델이 과부하일 때 다음 모델로 자동 전환한다. 세션이 끊기지 않고 이어진다.

© 2026 젠아이랩스 (GenAI Labs) · 강사 박수현 · v0.1.0 · c2b23b7

출처: Claude Code 공식 문서 | Anthropic Docs

첫 대화 5분

```
# 1. 리포 진입
cd ~/some-repo

# 2. Claude Code 시작
claude

# 3. 대화 (프롬프트가 뜨면 다음을 친다)
> @README.md 이 프로젝트를 3줄로 요약해줘
```

👀 관찰 포인트

- `@README.md` 는 파일 참조 문법이다. Claude가 그 파일을 읽어 컨텍스트에 얹는다.
- **Copilot과 완전히 다른 감각이다.** Copilot은 자동완성, Claude Code는 대화다.
- 파일 **읽기는 자동**, **쓰기·셸 실행은 승인 요청**이다. 이 지점에서 다음 파트인 모드로 이어진다.

🎯 첫 대화 추천 3

- `@README.md 3줄로 요약` — 컨텍스트 로딩 감각
- `!ls -la` — 셸 실행 감각
- `이 리포에서 pytest는 어떻게 실행하지?` — Claude가 스스로 파일을 뒤져 답을 찾는 감각

© 2026 젠아이랩스 (GenAI Labs) · 강사 박수현 · v0.1.0 · c2b23b7

PART 2

실행 모드 — *Shift+Tab으로 순환*

> 모드 6종 개요 · Default(Manual) · Plan · acceptEdits · `auto` · `dontAsk` · bypassPermissions · 상황별 모드 선택.

출처: Claude Code 공식 문서 | Anthropic Docs

실행 모드 6종 — *기본 순환 3 + 조건부 2 + 플래그 전용 1*

Claude Code의 실행 모드는 **"승인을 어디까지 물어보나"**를 정하는 스위치다. Shift+Tab 기본 순환은 `default → acceptEdits → plan` 셋이고, `bypassPermissions` · `auto`는 조건이 맞을 때 순환 뒤에 붙는다. `dontAsk`는 순환에 없이 플래그로만 켠다. 모두 합쳐 **6모드**다.

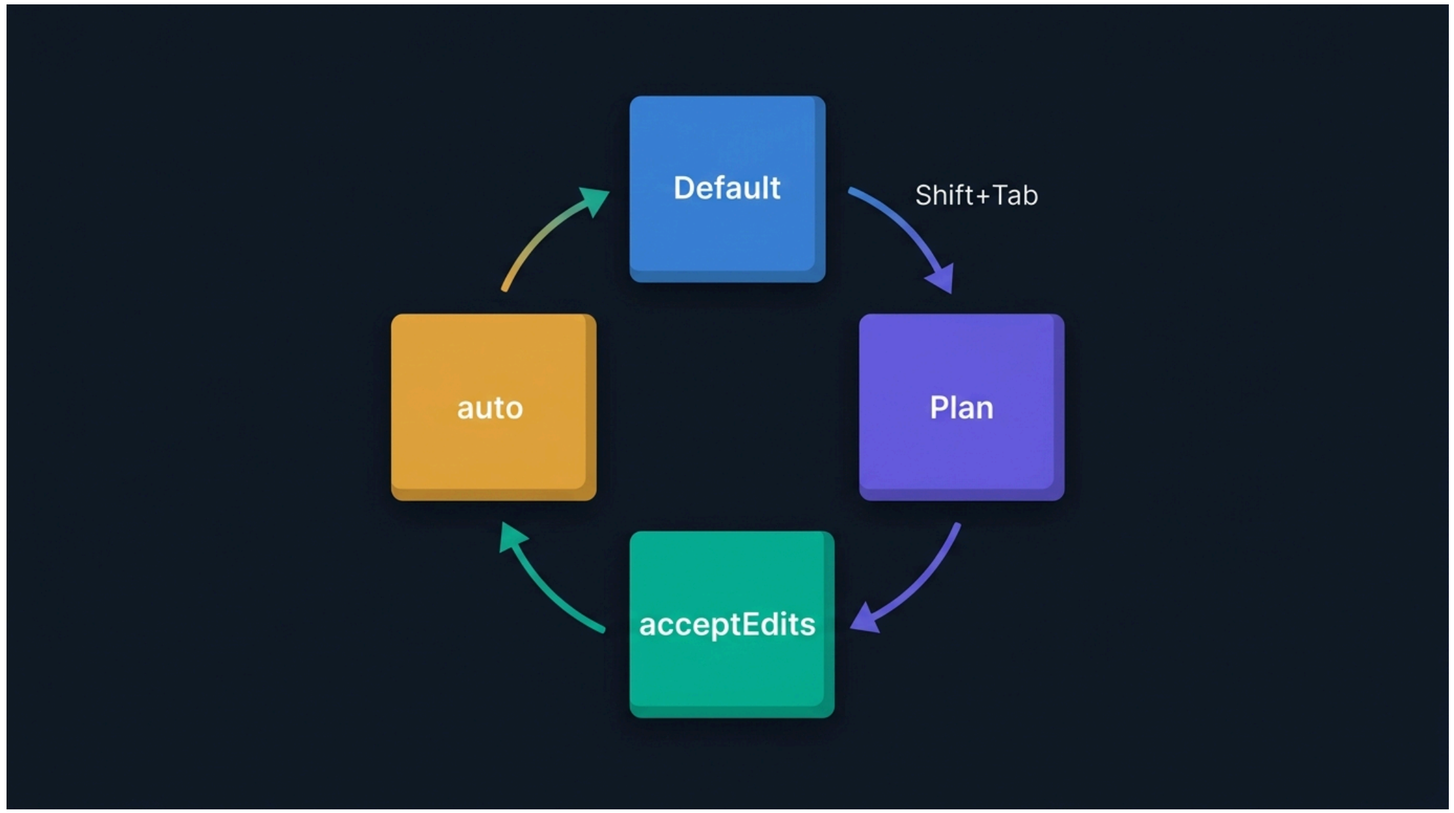
모드 (내부 키)	파일 편집	셸 명령	언제
Default *(CLI·IDE 표기 "Manual")*	승인 요청	승인 요청	처음·익숙하지 않은 리포·민감한 작업
Plan	**금지** (읽기만)	읽기 전용만	큰 리팩토링 시작 전·탐색
acceptEdits *(공식: Auto-accept edits)*	자동	여전히 승인	신뢰도 확보 후·반복 편집
`auto` *(2026 신규)*	자동	안전 분류기 통과 시 자동	Bypass 대신·긴 자동화 작업
`dontAsk` *(플래그 전용)*	사전 승인 도구만	사전 승인 도구만	잠금형 CI·스크립트
bypassPermissions	자동	자동	격리 환경(Docker·VM) 전용

📌 **핵심 개념.** 모드는 **"승인 정책"**을 바꾸는 것이지, **"AI의 능력"**을 바꾸는 게 아니다. 같은 작업을 하되, 어디서 사용자에게 물어볼지가 다르다.

📌 **2026년 추가된 `auto` 모드.** 승인은 건너뛰되 셸·네트워크 액션이 안전 분류기를 통과해야 실행된다. **안전 분류기 = 네트워크 도메인·명령 위험도·shell escape 사전 검사.** `bypassPermissions`보다 안전한 중간 지점이다.

📌 **`dontAsk`·Manual 라벨.** `default`는 CLI·IDE 확장에서 **Manual**로 표기(별칭 `manual`). `dontAsk`는 사전 승인 도구만 실행·나머지 자동 거부(잠금형 CI용). `--dangerously-skip-permissions` = `--permission-mode bypassPermissions`.

© 2026 젠아이랩스 (GenAI Labs) · 강사 박수현 · v0.1.0 · c2b23b7



실행 모드 — Shift+Tab으로 순환

© 2026 젠아이랩스 (GenAI Labs) · 강사 박수현 · v0.1.0 · c2b23b7

출처: Claude Code 공식 문서 | Anthropic Docs

Default 모드 — *안전 · 대화형*

설치 직후 기본값이다. Claude가 파일을 수정하거나 셸을 실행하려 할 때마다 사용자에게 승인을 묻는다.

✅ 언제 쓰나

- 새 리포에 처음 진입할 때·팀 컨벤션을 모를 때
- 익숙하지 않은 라이브러리·프레임워크
- 프로덕션 코드·실수 비용이 큰 상황

🔍 주의점

- Default에서도 파일 **읽기는 승인 없이 자동**이다
- 승인이 걸리는 것은 **쓰기·실행·삭제**다
- "Yes, and don't ask again for this session"을 남발하면 실질적으로 acceptEdits가 된다

📌 **감각 잡기 팁.** 처음에는 **매번 승인**을 유지한다. Claude가 어디에서 손을 대는지 눈으로 익힌 뒤에 acceptEdits로 넘어간다.

© 2026 젠아이랩스 (GenAI Labs) · 강사 박수현 · v0.1.0 · c2b23b7

출처: Claude Code 공식 문서 | Anthropic Docs

Plan 모드 — *읽기 전용 · 큰 리팩토링 전에*

Plan 모드는 **"연구하고 제안하되 손대지 않는"** 상태다. Claude는 파일을 읽고, 셸 탐색 명령을 돌리고, **계획을 글로 쓴다.** 실제 편집은 하지 않는다.

```
# 세션 시작 시
claude --permission-mode plan

# 세션 안에서
Shift+Tab (모드 순환)
```

🎯 언제 쓰나

- **큰 리팩토링 시작 전** — "인증 모듈을 세션 기반에서 JWT로 이관" 같은 대형 작업 계획
- **낯선 리포 탐색** — "이 리포 아키텍처를 3줄로 정리해줘"
- **아키텍처 결정** — 여러 접근을 비교하고 싶을 때

🔀 Plan 모드 종료

- 계획이 마음에 들면 `Shift+Tab`으로 Default나 acceptEdits로 넘어가 실행한다
- 계획이 마음에 안 들면 그대로 세션을 종료한다
- 계획 검토와 실행 전환의 흐름을 몸에 익힌다

© 2026 젠아이랩스 (GenAI Labs) · 강사 박수현 · v0.1.0 · c2b23b7

출처: Claude Code 공식 문서 | Anthropic Docs

acceptEdits 모드 — *편집 자동 · 셸은 승인*

Claude를 며칠 써 보면 승인 프롬프트가 반복 노동이 된다. 신뢰가 쌓이면 자동화 모드로 넘어간다. 그 첫 단계가 acceptEdits — **공식 UI 명칭은** `Auto-accept edits`, CLI·설정 파일에서는 `acceptEdits` 키로 다룬다.

⚙️ 자동 vs 승인

- 파일 편집 → **자동**
- 기본 파일시스템 명령 → **자동**
- 셸 명령 → **여전히 승인 요청**

안전과 속도의 균형점.

✅ 언제 쓰나

- 자기 리포·익숙한 코드베이스
- 반복 편집 (린트 정리, 형식 통일 등)
- 신뢰도가 확보된 태스크

```
# 진입 방법
Shift+Tab (모드 순환)
# 또는
claude --permission-mode acceptEdits
```

📌 acceptEdits는 대부분의 개인 개발자가 며칠 안에 정착하는 모드다. 셸 명령까지 자동화하고 싶으면 다음 슬라이드의 `auto` 나 `bypassPermissions` 로 넘어간다.

© 2026 젠아이랩스 (GenAI Labs) · 강사 박수현 · v0.1.0 · c2b23b7

출처: Claude Code 공식 문서 | Anthropic Docs

bypassPermissions — ⚠️ 격리 환경 전용

🚨 **개인 로컬 · 프로덕션 노트북에서 절대 사용 금지 — 실수 한 번에 홈 디렉토리가 날아간다.**

```
claude --dangerously-skip-permissions
```

❌ 위험 지점

- 승인 프롬프트와 안전 검사를 함께 끈다 — 파일·셸 모두 즉시 실행
- **보호 경로(`.git` · `.zshrc` · `.claude`) 쓰기도 허용**된다 (최근 버전부터)
- 완전 자동화는 곧 완전 실수 자동화다

✅ 오직 여기서만

- Docker · WSL 컨테이너
- CI 러너 · 격리 VM

공통점: 날아가도 5분이면 다시 세우는, 사람 자산이 없는 환경.

📌 **여전히 묻는 것은 셋뿐** — ① `rm -rf /` · `rm -rf ~` **(모델 오작동 대비 서킷 브레이커)** ② 명시한 `ask` 규칙 ③ `requiresUserInteraction` MCP 도구. **보호 경로가 지켜지는 모드는 Default·Plan·acceptEdits 다.**

📌 **더 안전한 대안 —** `--permission-mode auto`**.** 승인 없이 진행하되 셸·네트워크가 안전 분류기를 통과해야 실행된다.

© 2026 젠아이랩스 (GenAI Labs) · 강사 박수현 · v0.1.0 · c2b23b7

출처: Claude Code 공식 문서 | Anthropic Docs

상황별 모드 선택 — *5개 시나리오*

상황	권장 모드	이유
처음 만난 리포·코드 이해	**Plan**	손대지 않고 파악한다
익숙한 리포·새 기능 개발	**Default**	매 편집 확인·방향 통제
자기 사이드 프로젝트·반복 편집	**acceptEdits**	편집 자동·셸만 승인
CI·Docker 컨테이너 안 배치 작업	**bypassPermissions or auto**	사람이 승인할 수 없는 환경
대형 리팩토링 (인증·DB 마이그레이션)	**Plan → Default**	계획 확정 후 실행 전환

📌 이 표를 몸에 익히면 승인 프롬프트에 시간을 뺏기지 않는다. **벽에 붙여 두고 쓰도록 만든 표다.** 스크린샷을 찍어 둔다.

© 2026 젠아이랩스 (GenAI Labs) · 강사 박수현 · v0.1.0 · c2b23b7

PART 3

슬래시 명령어 총정리 — *5개 그룹*

> 슬래시 명령이 뭔가 · 그룹 A~E · 요약 카드. 20개 넘는 명령을 한꺼번에 외우지 않고 그룹별로 익힌다.

출처: Claude Code 공식 문서 | Anthropic Docs

슬래시 명령이 뭔가

슬래시 명령은 세션 안에서 `/`로 시작하는 특수 명령이다. **자연어 대화와 별개로 세션 자체를 조작**하는 데 쓴다.

🔍 전체 목록 보는 방법

- 세션 안에서 `/`만 치면 자동완성이 뜬다
- 또는 `/help`를 치면 전체 목록이 나온다
- 20개 넘는 명령을 통째로 외우지 않는다

📚 다섯 그룹으로 나눠 익힌다

1. **세션·컨텍스트 관리** — 대화 흐름 조작
2. **프로젝트·설정** — 리포·모델·설정
3. **확장·통합** — MCP·훅·스킬
4. **진단·유틸** — 문제·비용·도움
5. **Git·PR** — 저장소 통합

📌 대화 초기화·모델 전환·진단·설정·MCP 관리는 모두 슬래시 명령이다. 자연어로 "대화 초기화해줘"라고 시키지 않는다. `/clear`를 친다.

© 2026 젠아이랩스 (GenAI Labs) · 강사 박수현 · v0.1.0 · c2b23b7

출처: Claude Code 공식 문서 | Anthropic Docs

그룹 A — *세션·컨텍스트 관리*

명령	하는 일	언제 쓰나
`/clear`	대화 이력·컨텍스트를 전체 초기화	완전히 다른 태스크로 전환할 때·파일 편집은 남는다
`/compact`	이전 대화를 요약본으로 압축	컨텍스트 한도 근접·대화 흐름은 유지
`/resume`	이전 세션 이력에서 선택해 재개	어제 하던 작업으로 돌아가고 싶을 때
`/todo`	세션의 to-do 리스트 관리	Plan 모드 이후·큰 작업 추적
`/init`	현재 디렉토리에 `CLAUDE.md` 자동 생성	새 리포에 처음 진입할 때

📌 **핵심 대비 — `/clear` vs `/compact`.** - `/clear` — 이력을 완전히 삭제한다. 컨텍스트 창이 텅 빈다. **다른 태스크로 넘어갈 때 쓴다.** - `/compact` — 요약본으로 압축한다. 대화 맥락은 유지하고 토큰만 절약한다. **같은 태스크를 계속할 때 쓴다.**

📌 **`/init`의 실제 효과.** 리포 구조·기존 코드·git 이력을 훑고, 팀·프로젝트 관례를 담은 `CLAUDE.md`를 자동 생성한다. **이 파일은 초안이므로 자기 규약에 맞게 편집한다.**

📌 **자동 컴팩트(auto-compact).** 컨텍스트 창이 한계에 근접하면 이전 대화가 **자동으로 요약·압축**된다 — 수동 `/compact` 와 같은 동작이 알아서 일어난다. 켜고 끔은 `/config` 의 Auto-compact 항목(설정 `autoCompactEnabled`, 기본 켜짐)으로 조정한다.

© 2026 젠아이랩스 (GenAI Labs) · 강사 박수현 · v0.1.0 · c2b23b7

출처: Claude Code 공식 문서 | Anthropic Docs

그룹 B — *프로젝트·설정*

명령	하는 일	언제 쓰나
`/memory`	개인 memory 파일 편집	자주 반복하는 지침을 저장한다
`/config`	세션 설정 UI 열기	자동 업데이트 채널·테마·알림
`/model`	모델 전환	어려운 작업 → Opus, 반복 작업 → Haiku
`/agents`	subagent 관리·대시보드	Task로 띄운 병렬 세션 확인
`/status`	현재 세션 상태 요약	모델·모드·MCP·훅·도구 확인

🎯 `/model` 실전 팁

세션 중 어려운 결정 지점에서 잠깐 Opus로 올렸다가, 반복 작업으로 넘어가면 다시 Sonnet·Haiku로 내리는 게 표준이다. **요금·응답 속도가 즉시 반영된다.**

📊 `/status`는 진단의 첫 명령

세션 시작 직후 한 번 쳐서 "지금 어떤 상태인지" 확인한다. 여러 리포를 오가며 작업할 때 특히 필수다.

© 2026 젠아이랩스 (GenAI Labs) · 강사 박수현 · v0.1.0 · c2b23b7

출처: Claude Code 공식 문서 | Anthropic Docs

그룹 C — *확장·통합*

명령	하는 일	관련 축
`/mcp`	연결된 MCP 서버 목록·관리	MCP 축
`/hooks`	훅 설정 (PreToolUse·PostToolUse 등)	하네스 축
`/permissions`	허용·거부 도구 규칙 편집	하네스 축
`/statusline`	하단 상태표시줄 커스터마이즈	하네스 축
`/plugins`	플러그인 설치·관리	(선택)
`/agents`	subagent 정의 편집	컨텍스트 축

📌 **이 그룹은 컨텍스트·하네스·MCP 축에 걸친 명령이다. 여기서는 이름만 심고, 필요한 시점에 자세히 다룬다.**

빠르게 훑는 이유 — 셋업 세션에서 전부 만지려 하면 시간이 안 맞고 개념 밀도도 낮아진다. "무엇이 있다"만 알아두면 된다.

© 2026 젠아이랩스 (GenAI Labs) · 강사 박수현 · v0.1.0 · c2b23b7

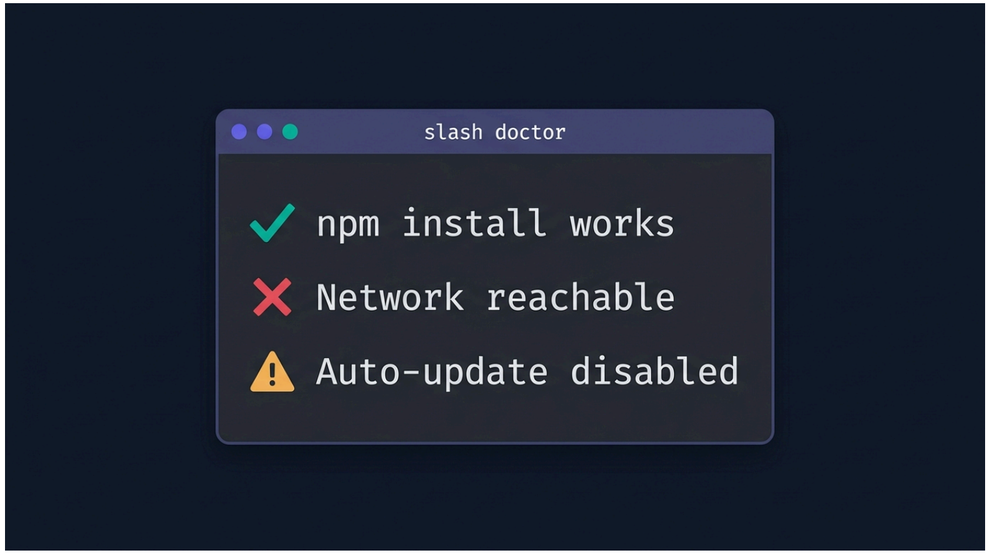
출처: Claude Code 공식 문서 | Anthropic Docs

그룹 D — *진단·유틸*

명령	하는 일	언제 쓰나
`/doctor`	환경 진단·오류 리포트	뭔가 이상할 때·업그레이드 직후
`/cost`	현재 세션 사용 비용·토큰	비용 걱정·한도 확인
`/bug`	버그 리포트를 Anthropic으로 제출	재현되는 버그를 발견했을 때
`/help`	슬래시 명령 전체 목록	명령이 기억 안 날 때
`/release-notes`	최근 업데이트 내역	새 기능이 궁금할 때
`/vim`	Vim 키바인딩 모드 토글	Vim 사용자용

🏥 `/doctor`는 트러블슈팅의 첫 명령

로그인 실패·MCP 연결 오류·업데이트 실패·ripgrep 미탐지를 진단하고 조치를 알려준다. 설치된 **Skill·플러그인·MCP 서버 목록**도 함께 보여줘, 지금 무엇이 로드돼 컨텍스트를 차지하는지 한눈에 확인한다.



© 2026 젠아이랩스 (GenAI Labs) · 강사 박수현 · v0.1.0 · c2b23b7

출처: Claude Code 공식 문서 | Anthropic Docs

그룹 E — *Git · PR*

명령	하는 일	언제 쓰나
`/review PR#`	특정 PR을 Claude가 리뷰한다	팀 리뷰·자기 PR 셀프 리뷰
`/pr-comments`	현재 PR의 리뷰 코멘트 읽기	PR 응답·수정 반영

🤖 실전 활용 — CI 자동화

CI 안에서 `claude -p "/review 123"` **형태로 자동화**하면 PR이 열릴 때마다 Claude가 자동 리뷰를 남긴다. 팀 도입 초기에 가장 먼저 성과가 나오는 조합이다.

⚠️ 주의

`/review`는 GitHub CLI(`gh`)가 인증된 상태를 전제로 한다. `gh auth login`이 안 돼 있으면 실패한다.

📌 이 그룹은 두 명령뿐이다. **명령 이름만 심어두고**, 실전 CI 통합·PR 리뷰어는 실습에서 반복 사용한다.

© 2026 젠아이랩스 (GenAI Labs) · 강사 박수현 · v0.1.0 · c2b23b7

출처: Claude Code 공식 문서 | Anthropic Docs

슬래시 명령 요약 카드 — *벽에 붙이기*

상황	명령
뭔가 이상하다	`/doctor`
비용이 걱정된다	`/cost`
컨텍스트가 넘칠 것 같다	`/compact`
완전히 다른 태스크로	`/clear`
새 리포 첫 진입	`/init` → `/status`
모델을 바꾸고 싶다	`/model opus` (또는 sonnet·haiku)
MCP 서버 확인	`/mcp`
PR 리뷰	`/review 123`
어제 하던 작업으로	`/resume`

📌 **스크린샷 한 장으로 벽에 붙이는 카드.** 참여자가 실전에서 가장 자주 다시 찾는 자료다.

© 2026 젠아이랩스 (GenAI Labs) · 강사 박수현 · v0.1.0 · c2b23b7

PART 4

CLI 인자·특수 입력 — *세션 밖에서*

> `claude` 5개 형태 · 실전 플래그 7선 · `@file` · `!command` · `Shift+Tab` · `Esc`.

출처: Claude Code 공식 문서 | Anthropic Docs

`claude` CLI 5개 핵심 형태

형태	하는 일	예시
`claude`	인터랙티브 세션 시작	`claude`
`claude "질문"`	첫 프롬프트를 넣고 인터랙티브 시작	`claude "이 리포 요약해줘"`
`claude -p "질문"`	프롬프트 실행 후 결과만 출력하고 종료(배치)	`claude -p "@README.md 요약"`
`claude -c`	이 디렉토리의 최근 세션 이어서 재개	`claude -c`
`claude -r "세션명"`	특정 세션 ID·이름으로 재개	`claude -r "auth-refactor"`

📌 `-p` **(print mode)의 핵심.** 세션이 아니라 **일회성 응답** 을 받는다. 셸 파이프·CI·스크립트에 쓴다.

© 2026 젠아이랩스 (GenAI Labs) · 강사 박수현 · v0.1.0 · c2b23b7

출처: Claude Code 공식 문서 | Anthropic Docs

`-p` 실전 — *셸 파이프·CI·스크립트*

```
# 셸 파이프 - 로그 파일 요약
cat production.log | claude -p "이 로그의 에러 패턴 5가지"

# @file 참조 - 배치
claude -p "@README.md 한 줄 요약해줘"

# CI 안에서 - PR 리뷰
claude -p "/review 123" --model haiku
```

🔧 개인 alias로 도구화

자기 셸 rc에 alias 하나만 추가해도 개인 도구가 된다.

```
# ~/.zshrc
claude-summary() {
  claude -p "$@ 요약해줘"
}
```

📦 CI·자동화 감각

- 세션이 안 뜨고 결과만 stdout으로 나온다
- 셸 파이프·GitHub Actions·cron 어디든 붙는다
- `--output-format json`으로 스크립트 파싱까지 가능하다

📌 **자동화 진입점.** `-p`가 손에 익으면 이후 CI 통합·PR 리뷰어·MCP 서버 실습이 매끄럽게 이어진다.

© 2026 젠아이랩스 (GenAI Labs) · 강사 박수현 · v0.1.0 · c2b23b7

출처: Claude Code 공식 문서 | Anthropic Docs

자주 쓰는 CLI 플래그 7선

플래그	하는 일	예시
`--model`	세션 모델 지정	`--model opus`
`--permission-mode`	시작 시 모드 지정	`--permission-mode plan`
`--add-dir`	다른 디렉토리도 접근 허용	`--add-dir ../shared-libs`
`--dangerously-skip-permissions`	bypassPermissions로 시작(격리 환경만)	`--dangerously-skip-permissions`
`--fallback-model`	과부하 시 대체 모델	`--fallback-model sonnet,haiku`
`--worktree` (`-w`)	git worktree 격리 세션	`-w feature-auth`
`--output-format json`	스크립트 파싱용 JSON 출력(`-p`와 함께)	`claude -p "..." --output-format json`

📌 `claude --help`는 전체 플래그를 보여주지 않는다(공식 문서에도 명시돼 있다). 이 7개가 실무에서 자주 쓰는 것들이다.

📌 **`--worktree` 배경 — `git worktree`.** 한 리포를 **두 벌 이상 병렬 체크아웃**해 브랜치별로 격리한 작업 트리를 만드는 기능이다. Claude Code에 `-w feature-auth`를 붙이면 해당 worktree 하위에서만 편집·셸이 돌아가 다른 브랜치와 충돌하지 않는다.

© 2026 젠아이랩스 (GenAI Labs) · 강사 박수현 · v0.1.0 · c2b23b7

출처: Claude Code 공식 문서 | Anthropic Docs

서브커맨드 3 + MCP 관련 4

🛠️ 기본 서브커맨드 3

```
claude --version   # 설치 버전
claude doctor      # 환경 진단
claude update      # 최신 버전으로 업데이트
```

- 인터랙티브 세션 없이 즉시 실행된다
- 셋업·업데이트 사이클에 쓴다

🔌 MCP 관련 서브커맨드

```
claude mcp add --transport stdio \
    my-server -- python server.py
claude mcp list
claude mcp login sentry
claude mcp logout sentry
```

- MCP 서버 등록·관리
- 실전에서 반복 사용

📌 `claude doctor`는 셋업 직후·업그레이드 직후에 반드시 한 번 돌린다. 세션 안의 `/doctor`와 같은 진단이지만 세션 밖에서도 부를 수 있다.

© 2026 젠아이랩스 (GenAI Labs) · 강사 박수현 · v0.1.0 · c2b23b7

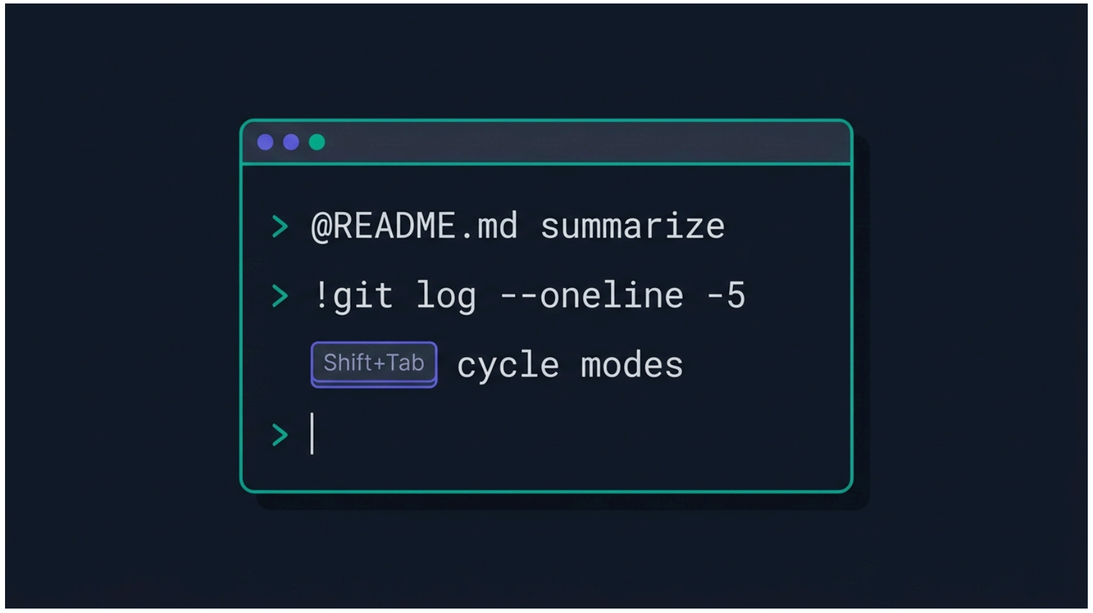
출처: Claude Code 공식 문서 | Anthropic Docs

특수 입력 4종 — *세션 안에서*

입력	하는 일	예시
`@file`	파일·폴더·URL을 컨텍스트에 로드	`@README.md` · `@src/` · `@https://...`
`!command`	셸 명령 실행 결과를 대화에 포함	`!git status` · `!ls -la`
`Esc`	진행 중인 응답 중단	(즉시 중단)
`Shift+Tab`	모드 순환	(Default → Plan → acceptEdits → ...)

📎 `@file` · 🐚 `!command`의 세부

- **파일 경로**는 리포 루트 기준 → `@src/main.py`
- **폴더 참조**는 폴더 전체를 훑는다 → `@tests/` (큰 폴더는 주의)
- **URL 참조**는 읽기 전용 → `@https://docs.python.org/...`
- `!` 는 프롬프트 맨 앞에 붙인다. 결과가 대화 컨텍스트에 자동으로 들어간다
- 예: `!pytest -x 실패한 테스트를 원인별로 분류해줘`



© 2026 젠아이랩스 (GenAI Labs) · 강사 박수현 · v0.1.0 · c2b23b7

나머지 유용한 단축키

⌨️ `Ctrl+C`

세션 종료 (2회 연속)

한 번 누르면 현재 입력을 취소한다. 두 번 연속으로 누르면 세션이 종료된다.

⬆️ `↑ / ↓`

이전 프롬프트 히스토리

셸 히스토리와 같은 감각이다. 방금 친 프롬프트를 편집해 재입력한다.

🧹 `Ctrl+L`

화면 클리어 (대화는 유지)

`/clear`와 다르다. 화면만 지우고 컨텍스트는 그대로 남는다.

📌 **`Ctrl+L`과 `/clear`를 헷갈리지 않는다.** - `Ctrl+L` — 화면만 지운다. 대화 컨텍스트는 유지된다. - `/clear` — 대화 컨텍스트를 완전히 삭제한다.

© 2026 젠아이랩스 (GenAI Labs) · 강사 박수현 · v0.1.0 · c2b23b7

PART 5

"언제 뭐 쓰나" 매트릭스 — *조합의 지도*

> 상황별로 조합한 매트릭스. 이 세션의 **최종 결과물**이다.

출처: Claude Code 공식 문서 | Anthropic Docs

상황·조합 매트릭스 ① — *셋업·개발 상황*

상황	조합
큰 리팩토링 시작 전	`--permission-mode plan` → `/todo`
반복 lint·정리	`--permission-mode acceptEdits` + `/model haiku`
컨텍스트가 넘칠 것 같다	`/compact` (맥락 유지) 또는 `/clear` (완전 리셋)
새 리포 첫 진입	`/init` → `/status` → CLAUDE.md 편집
뭔가 이상하다	`/doctor` → `/bug` → 재시작
PR 리뷰(개인)	`/review 123`

📌 이 6개 상황이 실무에서 가장 자주 온다. 손에 익으면 승인 프롬프트·기억 부담이 크게 줄어든다.

© 2026 젠아이랩스 (GenAI Labs) · 강사 박수현 · v0.1.0 · c2b23b7

출처: Claude Code 공식 문서 | Anthropic Docs

상황·조합 매트릭스 ② — *자동화·비용·격리*

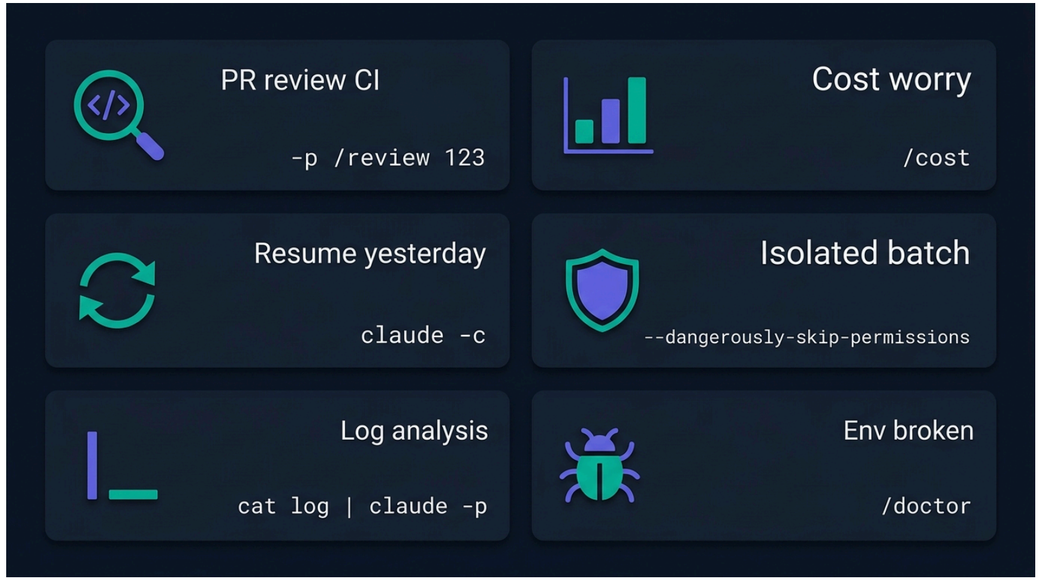
상황	조합
PR 리뷰(CI 자동)	`claude -p "/review 123" --model haiku`
비용이 걱정된다	`/cost` → `/model haiku`
어제 하던 작업으로	`claude -c` (이 디렉토리) 또는 `claude -r "이름"`
격리 환경에서 배치	`claude --dangerously-skip-permissions -p "..."` (Docker 안에서만)
로그 파일 분석	셸 파이프 + `-p` (아래 참고)

로그 파일 분석 — 셸 파이프

```
cat log | claude -p "에러 패턴 분류"
```

📌 이 5개는 **자동화·비용·격리**를 다룬다. 특히 `-p`와 셸 파이프 조합이 팀 도입에서 가장 먼저 성과가 나오는 지점이다.

⚠️ `--dangerously-skip-permissions` **행은 반드시 격리 환경 안에서만 쓴다.** 개인 로컬은 절대 금지.



© 2026 젠아이랩스 (GenAI Labs) · 강사 박수현 · v0.1.0 · c2b23b7

출처: Claude Code 공식 문서 | Anthropic Docs

오답 노트 — *참여자가 잘 저지르는 실수 5선*

실수	무엇이 문제	어떻게
개인 노트북에서 `--dangerously-skip-permissions`	홈 디렉토리·시스템 파일이 위험하다	격리 컨테이너 안에서만 쓴다
큰 리팩토링을 Default로 시작	매 편집 승인이 폭발한다·시간 낭비	Plan → 검토 → acceptEdits
`/clear`**를 남발**	대화 맥락을 통째로 잃는다	태스크 전환 시에만 쓰고, 같은 태스크에선 `/compact`
Opus로 반복 작업	요금·응답 속도 손해	반복은 Haiku로 전환한다
CLAUDE.md 없이 진입	Claude가 팀 관례를 모르고 매 세션 반복 지시가 필요하다	`/init`을 한 번 돌리고 이후 자기 규약에 맞게 편집한다

📌 경험 없는 참여자가 첫 주에 잘 저지르는 5개다. **미리 알아두면 이후에 헤매지 않는다.**

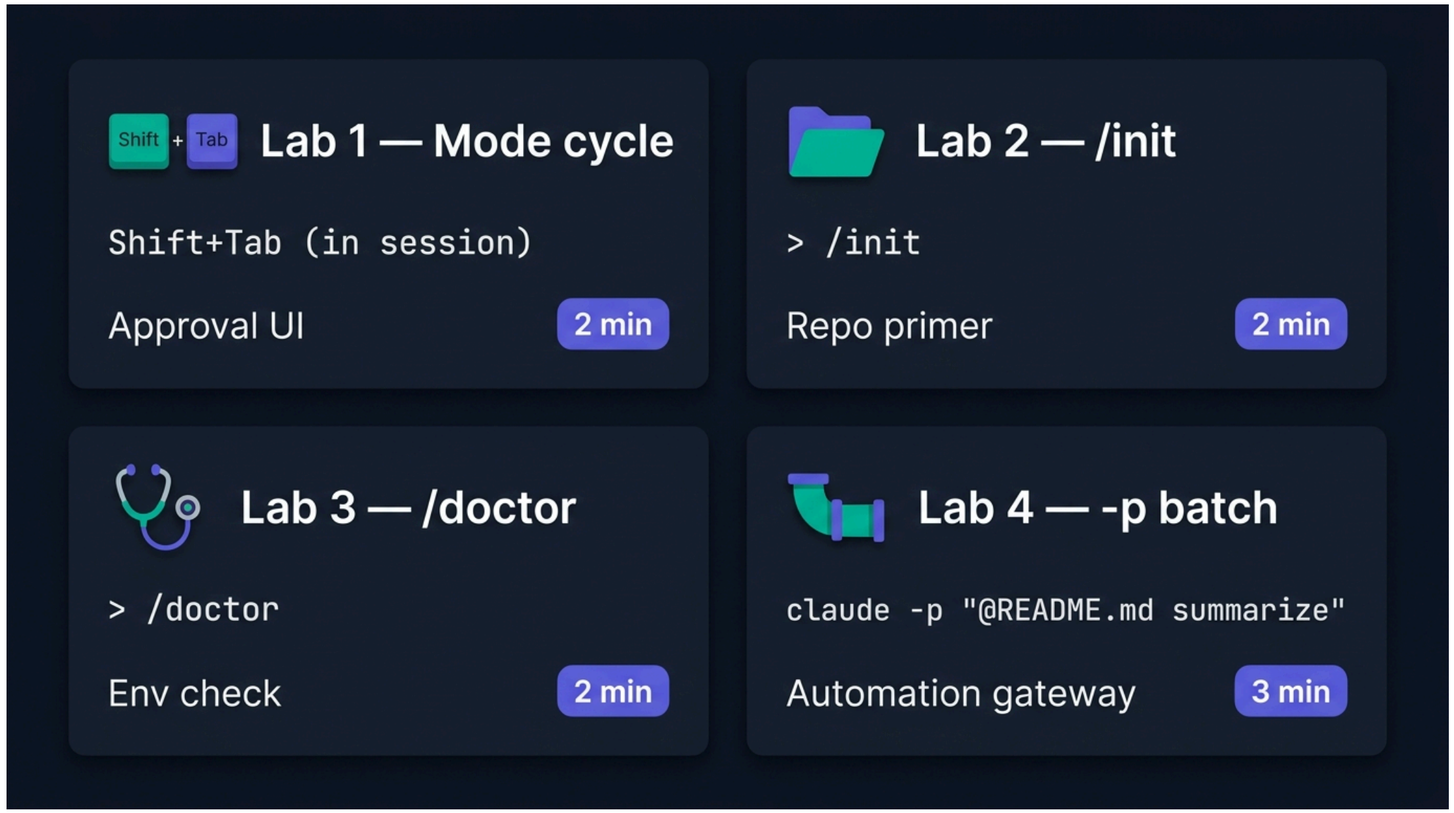
© 2026 젠아이랩스 (GenAI Labs) · 강사 박수현 · v0.1.0 · c2b23b7

PART 6

미니 실습·정리 — *손에 붙이기*

> 실습 4개 (모드 순환 · `/init` · `/doctor` · `-p` 배치) · 세션 정리 한 문장.

출처: Claude Code 공식 문서 | Anthropic Docs



미니 실습·정리 — 손에 붙이기

© 2026 젠아이랩스 (GenAI Labs) · 강사 박수현 · v0.1.0 · c2b23b7

출처: Claude Code 공식 문서 | Anthropic Docs

실습 1 — *모드 4개 순환*

```
# 1. 세션 시작
claude

# 2. Shift+Tab 을 4번 눌러 모드 순환 관찰
#    Default → acceptEdits → plan → (원래대로)
# 3. 각 모드에서 우상단 표시 확인

# 4. 각 모드에서 "test.txt 파일 만들어줘" 시도
#    - Default: 승인 요청 뜸
#    - Plan: "계획만 세우고 편집 안 함" 응답
#    - acceptEdits: 바로 만듦
```

📌 **관찰 포인트.** 같은 지시가 모드마다 다르게 반응한다. **승인 프롬프트 UI를 눈에 익히는 게 목표다.**

© 2026 젠아이랩스 (GenAI Labs) · 강사 박수현 · v0.1.0 · c2b23b7

출처: Claude Code 공식 문서 | Anthropic Docs

실습 2 — `/init` 실행

```
# 1. 자기 사이드 프로젝트 리포로 이동
cd ~/my-project

# 2. Claude Code 시작
claude

# 3. 세션 안에서
> /init
```

👀 관찰 포인트

자동 생성된 `CLAUDE.md`를 열어본다. 리포 개요·명령·관례가 담긴다.

📝 핵심 인식

이 파일은 초안이지 최종본이 아니다. 컨텍스트 규약을 세운 뒤 자기 규약에 맞게 편집한다.

📌 자기 리포에 `CLAUDE.md`가 이미 있다면 `/init`이 덮어쓰지 않도록 주의한다. 백업 후 실행한다.

© 2026 젠아이랩스 (GenAI Labs) · 강사 박수현 · v0.1.0 · c2b23b7

출처: Claude Code 공식 문서 | Anthropic Docs

실습 3 — `/doctor` 실행

```
# 세션 안에서
> /doctor
```

🔍 관찰 포인트

환경 진단 결과 형식을 읽는다. - `✓ passed` - `✗ failed` - `⚠ warning`

세 상태 아이콘의 의미를 눈에 익힌다.

🛠️ 실패 항목이 있으면

- 실제 조치까지 시도한다
- 예: 자동 업데이트 비활성화 → `/config`로 활성화
- **트러블슈팅 절차를 익히는 게 목표다**

📌 `/doctor`는 세션 안에서 하는 진단, `claude doctor`는 세션 밖에서 하는 진단이다. **결과는 동일**하지만 진입점이 다르다.

© 2026 젠아이랩스 (GenAI Labs) · 강사 박수현 · v0.1.0 · c2b23b7

출처: Claude Code 공식 문서 | Anthropic Docs

실습 4 — `-p` 배치 실행

```
# 셸에서 세션 없이 바로
claude -p "@README.md 한 줄로 요약해줘"
```

👀 관찰 포인트

- 세션이 안 뜨고 **결과만 stdout으로** 나온다
- 이 결과를 셸 파이프·alias·CI에 어떻게 붙일지 상상해본다
- **자동화 진입점이다**

🚀 확장 실습 (도전)

```
# 자기 셸 alias 만들기
alias summ='claude -p "요약해줘"'

# 로그 요약
cat /tmp/app.log | summ
```

개인 도구화의 첫 사례다.

📌 개인 도구화의 첫 사례다. 셸 alias 한 줄이 자동화 진입점이 된다.

© 2026 젠아이랩스 (GenAI Labs) · 강사 박수현 · v0.1.0 · c2b23b7

마무리 — *한 문장으로*

여기까지 마쳤다면 노트북에서 Claude Code가 뜨고, Shift+Tab으로 승인 모드를 오가며 6종 체계를 파악했고, 20개 넘는 슬래시 명령을 5개 그룹으로 파악했고, `claude -p`로 셸 파이프까지 붙인 상태다.

🎯 이 세션 요약 한 줄

"슬래시 명령은 세션 안, CLI 인자는 세션 밖. 승인 정책은 Shift+Tab."

상황이 애매하면 **세션 안인지 밖인지**, **어떤 모드인지**부터 자문한다.

🖥️ 세션 안

슬래시 명령 · `@file` · `!cmd` · `Shift+Tab`

⌨️ 세션 밖

`claude -p` · `--model` · `--permission-mode`

🎚️ 승인 정책

Default → acceptEdits → plan
(+auto·dontAsk·bypass)

© 2026 젠아이랩스 (GenAI Labs) · 강사 박수현 · v0.1.0 · c2b23b7