꼭 필요한

클로드 코드 명령어 모음과 유용한 팁

클로드 코드 명령어

터미널에서 실행하는 기본 명령어

☐ `claude`

클로드 코드를 실행합니다. 이 명령어 하나면 AI와 대화를 시작할 수 있습니다.

```
claude "〈 프롬프트를 입력하세요. 〉"
```

큰따옴표 안에 원하는 내용을 적으세요.
질문이나 요청과 함께 프로그램을 시작합니다.

☐ `claude -c`

가장 최근 대화를 이어서 계속합니다.

프로그램을 껐다 켜도 이전 작업을 바로 이어갈 수 있습니다.

☐ `claude update`

클로드 코드를 최신 버전으로 업데이트합니다.

새로운 기능을 사용하려면 정기적으로 업데이트하세요.

☐ `--model`

특정 AI 모델을 선택해서 시작합니다.

```
claude --model 〈 모델명 〉
```

세 가지 모델 중 하나를 선택할 수 있습니다.

- opus: 강력
- sonnet: 균형
- haiku: 빠름

□ `claude --debug` 디버그 모드로 실행합니다.

오류가 발생했을 때 자세한 정보를 보고 싶다면 이 옵션을 사용하세요.

세션 관리 명령어 04-2절

컴퓨터나 프로그램에서
사용자와 시스템이 연결되어 있는 일정한 기간

□ `/clear` 대화 내용을 모두 지우고 처음부터 다시 시작합니다.

새로운 주제로 작업하고 싶을 때 사용하세요.

□ `/rewind` 이전 대화 상태로 되돌립니다. 실수했거나

다른 방향으로 다시 시도하고 싶을 때 유용합니다.

□ `/compact` 대화 내용을 압축해서 정리합니다.

대화가 너무 길어져서 느려질 때 사용하면 좋습니다. 압축 수준과 범위를 세밀하게 조정할 수 있는 고급 옵션을 제공합니다.

- `/compact --level 1`: 불필요한 공백 및 중복을 제거
- `/compact --level 2`: 변수명 단축, 구조 단순화
- `/compact --level 3`: 핵심 로직만 유지

정보 확인 명령어

☐ `/help`

사용 가능한 명령어 목록과 간단한 설명을 보여줍니다.

Tab 키를 이용해 전체 안내(general), 명령어 목록(commands) 그리고 내가 만든 명령어(custom-commands) 중 원하는 항목을 선택해 확인할 수 있습니다.

☐ `/status`

현재 상태를 확인합니다.

사용 중인 모델, 버전, 계정 정보 등을 한눈에 볼 수 있습니다.

☐ `/todos`

현재 진행 중인 할 일 목록을 확인합니다.

클로드가 복잡한 작업을 수행할 때 자동으로 생성하는 체크리스트를 볼 수 있습니다.

☐ `/usage`

현재 플랜의 한도와 사용 현황을 확인합니다.

유료 플랜을 사용 중일 때 월별 한도를 체크할 수 있습니다.

현재 대화와 작업 정보를 저장 및 표시한 메모리 공간

☐ `/context`

현재 대화에서 사용 중인 컨텍스트 정보를 확인합니다.

얼마나 많은 내용을 다루고 있는지 알 수 있습니다.

설정 및 구성 명령어

☐ `/config`	설정 화면을 엽니다. 클로드 코드의 다양한 옵션을 변경할 수 있습니다.
☐ `/model`	AI 모델을 전환합니다. 작업의 복잡도에 따라 다른 모델을 선택할 수 있습니다.
☐ `/init`	현재 프로젝트에 CLAUDE.md 파일을 생성합니다. 이 파일에 프로젝트 설명, 코딩 규칙, 주의사항을 적어두면 클로드가 프로젝트를 더 잘 이해하고 도와줄 수 있습니다.
☐ `/add-dir`	추가 작업 디렉토리를 지정합니다. 여러 폴더에 걸친 프로젝트를 작업할 때 유용합니다.
☐ `/permissions`	접근 권한을 관리합니다. 클로드가 어떤 작업을 할 수 있는지 제어할 수 있습니다.
☐ `/terminal-setup`	터미널 설정을 조정합니다. Shift+Enter 키로 줄바꿈을 할 수 있게 하는 등의 편의 기능을 설정합니다.

계정 관리 명령어

☐ `/login` 앤트로픽 계정을 전환하거나 다시 로그인합니다.

☐ `/logout` 현재 계정에서 로그아웃합니다.

프로젝트 작업 명령어

☐ `/review` 코드 리뷰를 요청합니다. 작성한 코드의 문제점이나 개선점을 찾아줍니다.

☐ `/memory` CLAUDE.md 메모리 파일을 편집합니다.
프로젝트에 대한 클로드의 이해를 업데이트할 수 있습니다.

문제 해결 명령어

☐ `/doctor`

설치 상태를 진단합니다.

클로드 코드가 제대로 작동하지 않을 때 이 명령어로 문제를 찾을 수 있습니다.

☐ `/bug`

버그나 문제를 앤트로픽에 보고합니다.

예상치 못한 오류가 발생했을 때 사용하세요.

MCP 기본 명령어

☐ `/mcp`

MCP 서버 연결 상태를 확인하고 관리합니다. 08-1절

☐ `claude mcp list`

설정된 MCP 서버 목록을 보여줍니다.

☐ `@`

@ 기호로 특정 MCP 서버를 호출할 수 있습니다.

예를 들어, `@github`는 GitHub MCP 서버에게 도움말을 요청합니다. 04-3절

□ `claude mcp add`

원격 MCP를 추가하는 기본 명령 형태입니다.

```
claude mcp add --transport http <내가 정한 이름> <서버 URL>
```

로컬 MCP의 서버 URL

로컬 MCP를 추가하는 기본 명령 형태입니다.

```
claude mcp add 〈내가 정한 이름〉 〈옵션〉 -- 〈MCP 실행 명령〉
```

원격 MCP와 로컬 MCP — 08-1절

	원격 MCP	로컬 MCP
실행 위치	외부 서버	내 컴퓨터
난이도	간단	약간 복잡
속도	상대적으로 느림	빠름
보안	인증 필요	인증 불필요

프롬프트 모음

현장에서 바로 복사, 붙여넣기 해도 잘 독작하는 실전 프롬프트를 모았습니다.

Git 자동화 관련 프롬프트

깃에 코드를 업로드해 변경사항을 기록하고, 배포하기 위해 필요한 커밋, 푸시 등 동작도 직접 명령어를 입력하지 않고 요청하면 클로드 코드가 자동으로 처리해 줍니다. 역시 대화하듯 일상적인 자연어로 프롬프트를 입력하세요.

□ 변경 사항 반영

클로드 코드가 자동으로 변경된 파일을 확인하고, 적절한 커밋 메시지를 작성해서 커밋한 뒤 원격 저장소에 푸시합니다.

프롬프트

```
>_ 변경 사항을 커밋해줘.
>_ 변경 사항을 GitHub에 올려줘.
>_ 지금까지 작업한 내용을 커밋하고 푸시해줘
```

깃허브와 Git 명령어 이해하기 08-2절

- 추가(add): 원격 저장소를 만들고 준비 공간에 파일 추가
- 커밋(commit): 로컬 저장소에 파일 보관
- 푸시(push): 파일을 원격 저장소에 업로드

□ **브랜치 생성**

브랜치 생성부터 푸시까지 한 번에 처리합니다.

프롬프트

```
>_ feature 브랜치를 만들어 줘.
>_ 새 브랜치를 만들고 작업 내용을 푸시해 줘.
```

□ **풀리퀘스트 생성**

깃허브에 PR을 자동으로 생성하고, 요약과 테스트 계획까지 작성해 줍니다.

프롬프트

```
>_ Pull Request를 만들어 줘.
```

상황별 프롬프트 모음

□ **파일**

파일 생성, 수정, 탐색 등 기본적인 작업은 모두 자연어로 요청할 수 있습니다.

프롬프트

```
>_ index.html 파일을 만들어 줘.
>_ style.css에서 색상을 파란색으로 바꿔 줘.
>_ 모든 .js 파일을 찾아서 보여 줘.
```

□ **코드**

기능을 추가하거나 기존 코드를 개선하고 싶을 때도 복잡한 명령은 필요 없습니다. 원하는 목적을 문장으로 표현하면 클로드가 알아서 코드를 작성하고 수정해 줍니다.

프롬프트

```
>_ 로그인 기능을 추가해 줘.
>_ 이 함수를 더 효율적으로 만들어 줘.
>_ 에러 처리를 추가해 줘.
```

□ **설명 요청**

코드의 동작 원리나 오류 원인을 알고 싶을 때는 직접 물어보면 됩니다. 클로드는 코드 구조를 분석해 작동 방식과 문제점을 이해하기 쉽게 설명해 줍니다.

프롬프트

```
>_ 이 코드가 어떻게 작동하는지 설명해 줘.
>_ app.py 파일에 뭐가 있는지 보여줘.
>_ 왜 이 에러가 나는지 알려줘.
```

유용한 팁

토큰 관리

- AI가 텍스트를 처리할 때 사용하는 단위입니다.
- 영어 단어 하나가 1~2토큰, 한글은 글자당 1~2토큰 정도입니다.
- `/context` 명령어로 현재 사용 중인 토큰량을 확인하며 관리하는 것이 좋습니다.

비용 관리

- 긴 대화는 `/compact`로 압축하면 토큰을 절약할 수 있습니다.
- 기본 모델은 Sonnet이지만, 간단한 작업은 Haiku로, 복잡한 작업은 Opus로 전환할 수 있습니다.

효율적인 작업 방법

- 새 주제를 시작할 때는 `/clear`로 이전 대화를 정리하세요.
- 프로젝트 시작 시 `/init`으로 CLAUDE.md를 만들어 두면 클로드가 프로젝트를 더 잘 이해합니다.
- 실수했을 때는 `/rewind`로 되돌릴 수 있으니 부담 없이 시도해 보세요.

자주 발생하는 문제 해결

클로드 코드는 강력한 자동화 도구이지만, 실제로 사용하다 보면 예상치 못한 오류나 막히는 순간들을 자주 만나게 됩니다. 오류 메시지를 하나하나 해석할 필요 없이, 발생 원인과 해결책을 바로 찾을 수 있도록 구성했습니다.

로그인 오류	`/usage`로 토큰 만료 확인 후 다시 로그인하세요.
권한 오류	화면의 안내를 따라 필요한 권한을 승인하세요.
느린 응답	`/compact`로 대화를 압축하거나 `/clear`로 새로 시작하세요. 04-2절
파일을 찾지 못 함	터미널에서 올바른 폴더로 이동했는지 확인하세요.

상황별 클로드 코드 사용법

클로드 코드를 처음 접하는 독자라도 이 부록을 참고하면 상황별로 어떤 명령을 입력해야 하는지 쉽게 익힐 수 있습니다. 처음 프로젝트를 시작할 때부터 이전 작업을 이어 하거나 문제를 해결할 때까지, 자주 쓰이는 명령어와 대화 예시를 간단하게 정리했습니다. 실습 중 막히는 순간이 있을 때 이 페이지를 펼치면 곧바로 해결책을 찾을 수 있습니다.

시작할 때

클로드 코드를 처음 실행할 때는 기본 환경을 준비하고 프로젝트를 초기화하는 과정이 필요합니다. 아래 순서대로 따라 하면 손쉽게 첫 대화를 시작할 수 있습니다.

1. 프로젝트 폴더에서 마우스 오른쪽 버튼을 눌러 **터미널을 엽니다.**
2. `claude`를 입력해 **클로드 코드를 실행합니다.**
3. `/init`을 입력해 **프로젝트를 초기화합니다.**
4. **원하는 작업을 자연스럽게 요청합니다.**

작업을 이어서 할 때

이전에 진행하던 프로젝트가 있다면, 클로드 코드는 대화를 기억하고 바로 이어서 작업할 수 있습니다. 아래 절차를 따르면 중단했던 지점부터 자연스럽게 계속할 수 있습니다.

1. 터미널을 열고 **프로젝트 폴더로 이동합니다.**
2. `claude -c`를 입력해 **이전 대화를 이어갑니다.**
3. **필요한 작업을 계속 요청합니다.**

새로운 작업을 시작할 때

이전 대화 내용이 길어졌거나 전혀 다른 주제로 전환하고 싶을 때는 대화를 초기화하면 됩니다. 새로 시작할 준비가 되었다면 아래 단계를 따라 진행하세요.

1. 클로드 코드 실행 중 `/clear`를 입력합니다.
2. **새로운 주제로 작업을 시작합니다.**

비용을 확인할 때

클로드 코드는 사용량에 따라 토큰이 소모되므로, 정기적으로 비용을 확인하는 습관이 중요합니다. 아래 명령어로 간단히 사용 현황을 점검할 수 있습니다.

1. `/context`를 입력해 **현재까지 사용량을 확인합니다.**
2. `/usage`로 **플랜 한도를 확인합니다.**
3. 필요하다면 `/compact`로 **대화를 압축합니다.**

문제가 생겼을 때

작업 중 오류나 예상치 못한 문제가 발생하더라도 걱정하지 마세요. 클로드 코드에는 자동 진단과 문제 보고 기능이 있어 손쉽게 원인을 확인하고 해결할 수 있습니다.

1. `/doctor`를 입력해 **자동 진단을 실행합니다.**
2. 해결되지 않으면 `claude --debug`로 **자세한 정보를 확인합니다.**
3. 여전히 문제가 있을 경우 `/bug`로 **문제를 보고합니다.**