

바이브 메이커 · 무료 자료

ECC 플러그인, 설치 2줄과 처음 쓸 명령어

깃허브 별 30만 개를 받은 클로드코드 플러그인 ECC(Everything Claude Code). 설치 두 줄, 깔고 나서 처음 칠 명령어, 두 줄로는 안 깔리는 것, 지우는 순서까지 README 기준으로 정리했습니다.

설치 2줄 · 처음 쓸 명령어 10개 · 부탁 문장 · 지우는 순서

목차

한눈에	ECC 가 무엇이고, 무엇이 들어 있나	03
설치 전	준비물 둘, 그리고 공식 경로만	04
설치 2줄	클로드코드 안에서 두 줄	05
처음 쓸 명령어	314개를 다 볼 필요는 없습니다	06
부탁 문장	기능 하나를 끝까지, 세 번의 부탁	07
개발 밖 스킬	글 · 콘텐츠 · 조사에 쓰는 스킬	08
아껴 둔 한 가지	두 줄로는 안 깔리는 것 하나	09
겹침 · 지우기	겹쳐 깔지 않기, 그리고 지우는 순서	10
모음	붙여 넣는 것 한 장	11
막히면	증상 → 할 일	12

한눈에

ECC 가 무엇이고, 무엇이 들어 있나

ECC 는 클로드코드에 에이전트 · 스킬 · 명령어를 한 번에 넣어 주는 플러그인입니다. 옛 이름은 Everything Claude Code 이고, 저장소 이름이 ECC 로 바뀌었습니다.

71
에이전트
계획 · 리뷰 · 빌드 수리 · 보안 · 설계

314
스킬
테스트 · 조사 · 보안 · 문서 · 화면 · 데이터

98
명령어
스킬로 들어가는 짧은 입구

항목	값	근거
저장소	**`github.com/affaan-m/ECC`**	깃허브 · 2026-11-11
깃허브 별	**`300,626`**	깃허브 API · 2026-11-11
라이선스	**`MIT`**	저장소는 무료. 비공개 저장소용 깃허브 앱 ECC Pro 는 유료(README)
플러그인 이름	**`ecc@ecc`**	README
만든 사람	Affaan Mustafa	README: 2025년 9월 Anthropic x Forum Ventures 해커톤 우승 (공동)
일하는 순서	계획 → 테스트 → 구현 → 리뷰 → 검증 → 기억 → 개선 (README)	

> 숫자 71 · 314 · 98 은 2026-11-11 README 기준입니다. 저장소가 자주 바뀌어서 예전 글이나 영상의 숫자(64 · 261 · 84 등)와 다를 수 있습니다. 이 자료는 README 와 깃허브에서 확인한 것만 적었고, 직접 돌려 보지 않은 부분은 「README 기준」이라고 적었습니다.

설치 전

준비물 둘, 그리고 공식 경로만

설치 두 줄을 치기 전에 터미널에서 두 가지만 확인합니다.

복사해서 그대로 쓰세요

```
claude --version
git --version
```

확인	기준 (README)	안 되면
`claude --version`	클로드코드 2.1 이상	클로드코드를 최신으로 올린 뒤 다시
`git --version`	Git 이 깔려 있어야 함	Git 을 먼저 설치

공식 경로는 다섯 곳뿐입니다

README 맨 위 경고입니다. 아래 다섯 곳이 아닌 재업로드 · 비공식 미러는 프로젝트가 관리하지 않고, 악성 코드가 들어 있을 수 있다고 적혀 있습니다.

- 깃허브 저장소 `github.com/affaan-m/ECC`
- 플러그인 이름 `ecc@ecc`
- npm 패키지 `ecc-universal` · `ecc-agentshield`
- 깃허브 앱 `github.com/apps/ecc-tools`
- 사이트 `ecc.tools`

> 블로그나 영상에서 「이 파일을 받아서 넣으세요」라고 하면 위 다섯 곳인지부터 봅니다. 플러그인은 클로드코드 안에서 명령을 실행할 수 있는 훅을 같이 깝니다. 그래서 출처가 중요합니다.

설치 2줄

클로드코드 안에서 두 줄

터미널이 아니라 클로드코드 대화창에 칩니다. 한 줄씩 엔터.

첫 줄 저장소를 플러그인 마켓으로 추가

복사해서 그대로 쓰세요

```
/plugin marketplace add https://github.com/affaan-m/ECC
```

둘째 줄 ECC 플러그인 설치

복사해서 그대로 쓰세요

```
/plugin install ecc@ecc
```

README 는 이 경로로 스킬 · 에이전트 · 명령어 · 플러그인이 관리하는 훅이 설치된다고 적고, 바로 뒤에 이렇게 덧붙입니다. **여기서 멈추세요.** 그 위에 수동 설치를 또 얹지 말라는 뜻입니다.

깔렸는지 확인

클로드코드를 새로 연 뒤 아래를 칩니다. 목록에 `ecc@ecc` 가 켜져 있으면 끝입니다.

복사해서 그대로 쓰세요

```
/plugin list
```

두 줄이 낯설다면

README 가 먼저 권하는 것은 안내형 설치입니다. 터미널에서 `npx ecc-universal@2.2.3 setup` 을 치면 물어보며 깔아 줍니다(Node.js 18 이상 필요).

둘 중 하나만

안내형 설치와 위 두 줄은 같은 `ecc@ecc` 를 깝니다. 하나를 골랐으면 다른 하나는 하지 않습니다.

> 「이미 있다」「범위가 겹친다」 같은 오류는 클로드코드가 내는 것입니다. README 는 이때 수동 설치를 덧씌우지 말고, 안내형 설치로 정리하거나 겹친 범위를 먼저 풀라고 합니다.

처음 쓸 명령어

314개를 다 볼 필요는 없습니다

README 의 말 그대로입니다. 전체 목록 말고, 지금 하려는 일에서 시작하세요.

지금 하려는 일	여기서 시작 (README 표기 그대로)
기능 하나 만들기	`/ecc:plan "만들 기능 설명"` 다음에 `tdd-workflow` 스킬
버그 고치기	실패하는 테스트로 먼저 재현한 뒤 `tdd-workflow` 스킬
방금 쓴 코드 검토	`/code-review` (새 컨텍스트에서 다시 봄)
깃허브 이슈 살펴보기	`/ecc:issue-review 번호`
빌드가 깨졌을 때	`/build-fix`
안 쓰는 코드 정리	`/refactor-clean`
컨텍스트가 무거운지 보기	`/context-budget`
긴 세션을 끝낼 때	`/save-session` 또는 `/learn-eval`
나중에 이어서	`/resume-session`
에이전트 설정 점검	`/security-scan`

이름 앞의 ecc: 는 무엇인가

플러그인으로 깔면 명령어가 `/ecc:plan` 처럼 이름표가 붙은 꼴이 됩니다. 수동 설치에서는 `/plan` 처럼 짧은 꼴이 나올 수 있습니다(README). 위 표는 README 가 적은 표기를 그대로 옮긴 것이니, 내 화면에서 안 보이면 `/` 를 치고 뜨는 목록에서 고르면 됩니다.

> README 는 스킬이 주된 입구이고 명령어는 편한 입구로 남겨 둔 것이라고 적습니다. 명령어 이름은 판이 바뀌면 달라질 수 있습니다.

부탁 문장

기능 하나를 끝까지, 세 번의 부탁

README 의 「새 기능 시작」 흐름(계획 → 테스트 먼저 → 리뷰)을 우리말 부탁으로 옮겼습니다. 괄호 안만 내 일로 바꾸세요.

1 계획 만들기 전에 계획부터 받는다

복사해서 그대로 쓰세요

/ecc:plan "(예: 문의 폼을 만들고, 제출하면 내 메일로 알림이 오게)"

계획이 나오면 바로 만들게 하지 말고 읽습니다. 고칠 곳이 있으면 고쳐 달라고 한 뒤 다음으로 갑니다.

2 테스트 먼저 tdd-workflow 스킬을 부른다

복사해서 그대로 쓰세요

tdd-workflow 스킬로 진행해 줘. 방금 계획의 첫 단계부터, 실패하는 테스트를 먼저 쓰고 그 테스트가 통과할 만큼만 구현해.

3 리뷰 새 컨텍스트에서 다시 본다

복사해서 그대로 쓰세요

/code-review

README 는 결과가 코드만이 아니라고 적습니다. 계획, 실패한 테스트, 통과한 테스트, 리뷰 지적, 마지막 검증까지가 한 묶음의 증거입니다.

내보내기 전

복사해서 그대로 쓰세요

/security-scan

> 위 한국어 문장은 README 의 흐름을 옮겨 적은 것입니다. 명령어와 스킬 이름은 README 표기 그대로이고, 문장 자체는 직접 돌려 본 결과가 아닙니다. 스킬은 일이 필요할 때 불려 오므로(README), 이름을 문장에 넣어 주면 고르기 쉬워집니다.

개발 밖 스킬

글 · 콘텐츠 · 조사에 쓰는 스킬

대부분은 개발용이지만, 글쓰기와 조사 쪽 스킬도 들어 있습니다. README 가 설명을 적어 둔 것부터 봅니다.

스킬 이름	하는 일 (README 설명)
`article-writing`	내가 준 글투로 쓰는 긴 글. 흔한 AI 말투를 피한다
`content-engine`	여러 채널용 소셜 콘텐츠 만들기와 재가공
`market-research`	출처를 붙인 시장 · 경쟁사 · 투자자 조사
`frontend-slides`	HTML 슬라이드 만들기, PPTX 를 웹 발표로 옮기기
`investor-materials`	피치덱 · 한 장 요약 · 메모 · 재무 모델

저장소 skills 폴더에는 이 밖에 `marketing-campaign` · `seo` · `brand-voice` · `social-publisher` · `deep-research` · `video-editing` 같은 이름도 있습니다(2026-11-11 깃허브). 이쪽은 이름만 확인했고 내용은 열어 보지 않았습니다.

부탁 문장

복사해서 그대로 쓰세요

market-research 스킬로 조사해 줘. 주제는 (예: 동네 필라테스 스튜디오의 온라인 예약 서비스). 경쟁 서비스 다섯 곳을 표로, 줄마다 출처 주소를 붙여.

복사해서 그대로 쓰세요

content-engine 스킬로, 아래 글을 인스타 캡션 · 쓰레드 글 · 블로그 도입부 세 가지로 바꿔 줘. 내 말투는 붙여 넣은 예시를 따라.
(여기에 원문과 내 글 예시)

복사해서 그대로 쓰세요

article-writing 스킬로 써 줘. 주제는 (한 줄), 읽는 사람은 (누구). 아래 내 글 두 편의 말투를 그대로 따라.
(여기에 내 글 두 편)

> 「어디에 쓰든 다 된다」는 뜻이 아닙니다. README 가 적은 범위는 계획 · 리뷰 · 테스트 같은 개발 일이 중심이고, 위 스킬들은 그 옆에 같이 들어 있는 것입니다.

아껴 둔 한 가지

두 줄로는 안 깔리는 것 하나

설치 두 줄로 스킬 · 에이전트 · 명령어 · 훅은 들어옵니다. 그런데 「규칙(rules)」은 들어오지 않습니다.

README 문장입니다. 클로드코드 플러그인은 rules 를 배포할 수 없으니, 원하는 규칙 묶음만 직접 넣으라고 합니다. 규칙은 「항상 지킬 기준」이라 매번 읽히고, 그래서 고른 것만 넣는 쪽이 낫다고 적습니다.

터미널에서 공통 규칙 + 내 언어 하나만

복사해서 그대로 쓰세요

```
git clone https://github.com/affaan-m/ECC.git
cd ECC
mkdir -p ~/.claude/rules/ecc
cp -R rules/common ~/.claude/rules/ecc/
cp -R rules/typescript ~/.claude/rules/ecc/
```

마지막 줄의 `typescript` 는 내가 쓰는 것으로 바꿉니다. README 에 적힌 묶음은 `typescript` · `python` · `golang` · `swift` · `php` · `arkts` 입니다. 코드를 직접 다루지 않는다면 `rules/common` 한 줄만으로 충분합니다.

이것도 따로입니다 (README)	어떻게
MCP 서버	플러그인 설치는 묶음 MCP 를 자동으로 켜지 않는다. 필요하면 클로드코드의 `/mcp` 로 직접
`multi-` 로 시작하는 명령어	기본 설치에 들어 있지 않다. `ccg-workflow` 를 따로 깔아야 돈다
`/astra-review`	Codex CLI 가 깔려 있어야 한다. 부르면 내 코드 변경분이 OpenAI 로 간다

> 규칙 폴더는 파일을 하나씩 말고 폴더째 복사합니다(예: `rules/common`). 안의 파일끼리 서로를 가리키고 있어서입니다 (README).

겹침 · 지우기

겹쳐 깔지 않기, 그리고 지우는 순서

README 가 가장 여러 번 되풀이하는 말입니다. 한 도구에는 한 가지 방법으로만 설치하세요.

이러면 두 번씩 돕니다

플러그인을 깐 뒤 수동 전체 설치(`./install.sh --profile full`)를 또 하면 스킬 · 명령어 · 훅이 두 벌이 됩니다. 훅 파일을 `settings.json` 에 복사해 넣어도 같습니다.

이건 괜찮습니다

클로드코드에 한 번, Codex 에 한 번처럼 서로 다른 도구에 한 번씩 까는 것은 겹침이 아닙니다.

지우는 순서 (README)

1 **클로드코드에서 플러그인을 지운다**

두 줄로 깔았다면 대부분 여기서 끝납니다.

2 **설치 프로그램으로 깐 적이 있으면, 그 폴더에서 ECC 제거 명령**

지우기 전에 `--dry-run` 으로 무엇이 지워질지 먼저 봅니다.

3 **직접 복사한 규칙 폴더를 지운다**

앞 장에서 넣은 `~/.claude/rules/ecc` 입니다. 필요 없는 것만.

4 **다시 깔 거면 한 가지 방법으로 한 번만**

1 플러그인 지우기 터미널에서 (클로드코드 도움말 기준)

복사해서 그대로 쓰세요

```
claude plugin uninstall ecc@ecc
claude plugin marketplace remove ecc
```

2 설치 프로그램으로 깐 것 깔았던 폴더에서 (README)

복사해서 그대로 쓰세요

```
npx ecc-universal@2.2.3 uninstall --dry-run
npx ecc-universal@2.2.3 uninstall
```

> 1번의 두 명령은 README 가 아니라 클로드코드 도움말(`claude plugin --help` · 2026-11-11)에 나오는 제거 명령이고, 실제로 지워 보지는 않았습니다. README 는 「클로드코드에서 플러그인을 제거하라」고만 적습니다.

모음

붙여 넣는 것 한 장

이 쪽만 열어 두고 필요할 때 복사하세요.

설치 클로드코드 대화창에서, 한 줄씩

복사해서 그대로 쓰세요

```
/plugin marketplace add https://github.com/affaan-m/ECC
```

복사해서 그대로 쓰세요

```
/plugin install ecc@ecc
```

확인 클로드코드를 새로 연 뒤

복사해서 그대로 쓰세요

```
/plugin list
```

계획 만들기 전에

복사해서 그대로 쓰세요

```
/ecc:plan "(만들 기능을 한두 줄로)"
```

테스트 먼저

복사해서 그대로 쓰세요

```
tdd-workflow 스킬로 진행해 줘. 방금 계획의 첫 단계부터, 실패하는 테스트를 먼저 쓰고 그 테스트가 통과할 만큼만 구현해.
```

리뷰 다 만든 뒤

복사해서 그대로 쓰세요

```
/code-review
```

> 점검 `/security-scan` · 무거워졌을 때 `/context-budget` · 지우기는 터미널에서 `claude plugin uninstall ecc@ecc`

막히면

증상 → 할 일

증상	할 일 (README 기준)
둘째 줄에서 「이미 있다」 · 범위가 겹친다는 오류	수동 설치를 덧씌우지 않는다. 안내형 설치(`npx ecc-universal@2.2.3 setup`)로 정리하거나, 겹친 플러그인 범위를 먼저 푼다.
ECC 가 두 번 보이거나 훅이 두 번 돈다	대개 플러그인 위에 전체 수동 설치를 얹은 경우다. 「지우는 순서」 네 단계를 따른 뒤 한 가지 방법으로 다시 깐다.
「Duplicate hooks file」 오류	플러그인 훅은 클로드코드 2.1 이상이 알아서 읽는다. 플러그인의 `plugin.json` 에 hooks 칸을 직접 넣었을 때 나는 오류라, 그 칸을 뺀다.
명령어가 영상과 다르게 보인다	플러그인 설치는 `/ecc:plan` 꼴, 수동 설치는 `/plan` 꼴일 수 있다. `/` 를 치고 뜨는 목록에서 고른다.
깔고 나서 대화가 금방 무거워진다	`/context-budget` 으로 본다. 안 쓰는 규칙을 빼고, 안 쓰는 MCP 서버를 끈다. 플러그인은 깔린 목록 전체를 모델에 알리므로, 자리가 중요하면 골라 까는 설치를 쓰라고 README 가 적는다.
숫자가 이 자료와 다르다	정상이다. 71 · 314 · 98 과 별 300,626 은 2026-11-11 값이다. 지금 값은 저장소 README 에서 본다.
윈도우에서 일부 기능이 안 된다	README 는 윈도우에서 몇몇 선택 기능에 제한이 있다고 적는다. 기본 설치와 명령어는 윈도우 · 맥 · 리눅스 모두 지원한다.
그래도 안 풀린다	저장소의 설치 문제 양식(Issues → install problem)에 적는다. ECC 는 진단 정보를 자동으로 올리지 않는다.

> 더 묻고 싶으면 네이버 카페 「바이브메이커」 자료실 글 댓글로.