

전자책 Vol.3 · 설치/운영 가이드 · 2026년 10월 무료 공개판

폰으로 내 컴퓨터 AI 부리기

Claude·Codex·Grok·Cursor 연결부터 첫 결과 확인까지

마이너스베타스튜디오 · 2026년 10월 무료 공개판

목차

1. 서문	3
2. 1장. 텔레그램 브릿지가 푸는 문제	5
3. 2장. 왜 공식 MCP나 공식 앱만으로 끝내지 않는가	7
4. 3장. 공통 준비: 봇, chat_id, tmux, 토큰 경계	11
5. 4장. Claude Telegram Bridge 설치하기	14
6. 5장. Codex를 연결해 첫 답변 받기	18
7. 6장. 안전하게 운영하기	22
8. 7장. 문제 해결과 다음 단계	26
9. 8장. Grok·Cursor를 선택한 독자의 첫 연결	29
10. 템플릿 1. 설치 전 준비 시트	33
11. 템플릿 2. 토큰 안전 체크리스트	34
12. 템플릿 3. Claude 브릿지 설정 예시	35
13. 템플릿 4. Codex 브릿지 설정 예시	36
14. 템플릿 5. 운영 health check	37
15. 템플릿 6. 공식 앱·MCP·브릿지 선택표	38
16. 맺는글	39
17. 저자소개	39

서문

AI 코딩 도구를 실제 업무에 붙여 쓰기 시작하면 제일 먼저 부딪히는 벽은 모델 성능이 아니다. 자리에 앉아 있을 때는 터미널에서 바로 지시하면 된다. 문제는 이동 중, 침대에 누웠을 때, 다른 컴퓨터 앞에 있을 때, 긴 작업이 돌아가는 동안 잠깐 확인만 하고 싶을 때다.

이 책은 그 문제를 풀기 위해 만든 공개 도구를 기준으로 쓴다. 중심은 두 개다.

- `claude-telegram-bridge`: 이미 켜 둔 Claude Code 터미널 세션을 텔레그램에서 조종한다.
- `codex-telegram-bridge`: 이미 켜 둔 Codex CLI 세션을 텔레그램에서 조종한다.

Grok CLI나 Cursor CLI를 쓰는 독자를 위해 같은 구조의 Grok·Cursor 브릿지 첫 연결도 8장에서 다룬다.

네 도구의 목표는 같다. 텔레그램을 새 AI 채팅방으로 쓰는 것이 아니라, 내 컴퓨터에 떠 있는 실제 터미널 세션의 리모컨으로 쓰는 것이다. 그래서 작업 기록은 로컬 터미널과 세션 로그에 남고, 텔레그램은 입력·확인·승인·결과 수신 창이 된다.

이 책은 초보자를 위한 설치서이면서, 혼자 여러 AI 도구를 굴리는 사람을 위한 운영서다. BotFather에서 봇을 만들고, 한 명의 `chat_id`만 허용하고, `tmux` 세션을 띄우고, 브릿지를 붙이고, `/ping`과 일반 프롬프트로 확인하는 순서까지 다룬다. 따로 내려받을 실습 파일은 없다. 필요한 명령과 설정 예시는 모두 본문과 부록에 실려 있다. 다만 실제 토큰, 실제 `chat_id`, 실제 컴퓨터 이름은 절대 예시에 넣지 않는다. 당신 환경의 값은 당신 로컬 설정 파일에만 들어가야 한다.

책에서 쓰는 명령은 공개 저장소와 공개 패키지 기준이다. 2026년 10월 18일 기준으로 Claude 브릿지 **v0.14.8**, Codex 브릿지 **v0.9.19**, Cursor 브릿지 **v0.4.6**, Grok 브릿지 **v0.6.6**의 설치 방법과 설정 이름을 확인했다. 시간이 지나 버전이 바뀌면 README와 release note를 먼저 보고, 이 책의 구조를 현재 버전에 맞게 읽으면 된다.

PART

1부. 왜 브릿지가 필요한가

공식 앱·MCP와 다르게 풀어야 하는 문제

1장. 텔레그램 브릿지가 푸는 문제

AI 코딩 도구를 처음 쓰면 보통 “답변을 얼마나 잘하나”만 본다. 조금 지나면 질문이 바뀐다. “이걸 내 작업 흐름 안에서 계속 굴릴 수 있나?”가 더 중요해진다.

터미널 기반 도구는 강력하다. Claude Code든 Codex CLI든 로컬 파일을 보고, 명령을 실행하고, 테스트를 돌리고, 긴 작업을 이어갈 수 있다. 반대로 터미널 기반이라는 점 때문에 불편하다. 터미널 앞에 없으면 입력하기 어렵고, 작업이 끝났는지 확인하기 어렵고, 승인 프롬프트가 떠 있으면 멈춘 채 기다린다.

텔레그램 브릿지는 이 사이를 잇는다.

텔레그램 메시지

- > 로컬 브릿지 데몬
- > tmux 안의 Claude Code 또는 Codex CLI
- > 세션 로그/터미널 상태 감시
- > 텔레그램 답장

핵심은 “AI를 새로 호출하는 것”이 아니다. 이미 켜 둔 인터랙티브 세션을 조종한다. 사용자는 로컬 터미널을 계속 원본으로 보고, 텔레그램은 원격 조종기처럼 쓴다.

브릿지가 필요한 순간

가장 흔한 순간은 세 가지다.

첫째, 긴 작업을 맡기고 자리를 비울 때다. 테스트가 오래 걸리거나 PDF 빌드, 리포트 생성, 여러 파일 검토가 이어질 때 폰에서 진행 상황을 확인하고 싶어진다. 브릿지가 없으면 원격 접속을 열거나, 돌아올 때까지 기다려야 한다.

둘째, 승인 프롬프트가 작업을 멈출 때다. Codex CLI는 명령 실행 전에 사용자 승인을 요구할 수 있다. 터미널 앞에 없으면 작업이 거기서 멈춘다. Codex 브릿지의 REPL 모드는 보이는 승인 선택지를 텔레그램 버튼으로 보내고, 선택한 값을 다시 터미널에 주입한다.

셋째, 이미지·파일·음성 같은 모바일 입력을 바로 작업 세션에 넣고 싶을 때다. 폰에 있는 스크린샷을 텔레그램으로 보내면 브릿지는 파일을 로컬에 저장하고, AI에게 `local_path` 맥락으로 전달한다. AI는 이미 작업 중인 로컬 repo 안에서 그 자료를 참고할 수 있다.

새 AI 챗봇이 아니라 원격 리모컨

브릿지를 “텔레그램용 챗봇”으로 생각하면 설계가 흐려진다. 챗봇은 보통 새 대화를 만든다. 브릿지는 반대다. 기존 터미널 세션을 유지한다. 이 차이가 크다.

기존 세션을 유지하면 다음 장점이 생긴다.

- 로컬 터미널 화면과 텔레그램 응답이 같은 작업을 가리킨다.

- 작업 디렉터리, git 상태, 테스트 결과가 로컬 환경 기준으로 유지된다.
- 터미널에서 직접 입력한 작업도 텔레그램으로 결과를 받을 수 있다.
- 승인·선택 프롬프트 같은 TUI 상호작용을 폰에서 처리할 수 있다.

대신 책임도 생긴다. 브릿지는 로컬 권한 경로다. 텔레그램 봇 토큰과 chat_id가 잘못 관리되면 다른 사람이 내 AI 세션에 말을 걸 수 있다. 그래서 브릿지 설치의 첫 원칙은 기능보다 경계다.

두 브릿지의 역할 차이

Claude 브릿지와 Codex 브릿지는 같은 문제를 풀지만 구조가 다르다.

구분	Claude Telegram Bridge	Codex Telegram Bridge
대상	Claude Code 인터랙티브 세션	Codex CLI 인터랙티브 세션
기본 방식	tmux Claude pane에 메시지 주입, transcript JSONL에서 최종답 추출	tmux Codex TUI에 메시지 주입, Codex JSONL/session 상태 감시
설치 성격	setup wizard가 token·hook·service를 설정, 수동 설정도 가능	setup wizard, doctor, service/watchdog 지원
주요 강점	Claude Code의 transcript와 SessionStart hook을 활용한 답변 매칭	승인 버튼, 선택 프롬프트, 미디어 전달, typing/progress/reasoning mirror
적합한 사용자	Claude Code를 이미 tmux로 쓰는 사용자	Codex CLI를 오래 켜 놓고 폰에서 조종하려는 사용자

둘을 한 프로그램으로 합치지 않는 이유도 여기 있다. 겉으로는 둘 다 “텔레그램 메시지를 터미널에 넣는다”처럼 보이지만, 세션 로그 구조, 승인 프롬프트, 상태 복구 방식이 다르다. 공통 추상화가 오히려 운영 리스크가 될 수 있다. 그래서 공개 repo도 Claude 전용, Codex 전용으로 분리되어 있다.

책에서 다루지 않는 것

이 책은 서버 호스팅이나 토큰 발급을 대신해 주는 방법을 다루지 않는다. 텔레그램 봇 토큰은 각자 BotFather에서 만들고, 로컬 설정 파일에만 넣는다. 예시에는 실제 토큰이나 실제 chat_id가 나오지 않는다.

또한 이 책은 “무조건 우회 권한으로 실행하라”는 책이 아니다. 로컬 AI 도구의 승인·샌드박스 정책은 자신의 업무 위험도에 맞게 정해야 한다. 브릿지는 입력 경로를 열어 주는 도구일 뿐, 권한 판단을 대신하지 않는다.

이 장의 결론

텔레그램 브릿지는 AI 모델을 바꾸는 기술이 아니라 운영면을 바꾸는 기술이다. 내 컴퓨터의 보이는 세션을 원본으로 유지하고, 폰을 원격 입력·확인·승인 장치로 붙인다. 이 관점만 잡고 설치하면 뒤의 설정들이 왜 필요한지 자연스럽게 이해된다.

2장. 왜 공식 MCP나 공식 앱만으로 끝내지 않는가

브릿지를 만들기 전에 가장 먼저 물어야 할 질문은 이것이다.

이미 공식 앱도 있고, Telegram MCP 같은 플러그인 방식도 있는데 왜 별도 브릿지가 필요한가?

정답은 “항상 브릿지가 더 좋다”가 아니다. 공식 앱이나 MCP가 맞는 상황이 분명히 있다. 다만 이 책의 목표, 즉 “내 로컬 터미널에 이미 떠 있는 Claude Code·Codex CLI 세션을 폰에서 조종한다”는 목적에는 브릿지가 더 정확하다.

MCP 텔레그램 플러그인이 잘하는 일

MCP 방식의 텔레그램 도구는 AI가 도구 호출로 메시지를 보내야 할 때 유용하다. 예를 들어 “결과를 텔레그램으로 알려줘” 같은 단방향 알림에는 잘 맞는다. AI 세션 안에서 도구를 호출하고, 텔레그램 Bot API로 메시지를 내보내는 구조다.

이 방식은 단순하고 좋다. 하지만 원격 리모컨으로는 한계가 있다.

질문	MCP 도구 방식	브릿지 방식
사용자가 폰에서 먼저 메시지를 보내 AI 세션을 깨울 수 있나	도구 설계에 따라 별도 입력 루프가 필요	getUpdates polling으로 기본 지원
이미 켜진 tmux 터미널 세션에 입력을 넣나	보통 도구 호출 중심	tmux pane에 직접 paste
로컬 터미널 transcript가 원본으로 남나	별도 대화/도구 로그가 섞이기 쉬움	기존 세션 로그를 계속 사용
승인/선택 프롬프트를 처리하나	터미널 UI 상태를 보기 어렵다	Codex 브릿지는 visible prompt를 버튼으로 중계
중복 답장 위험을 제어하나	MCP reply tool과 다른 미러가 동시에 보내면 중복 가능	브릿지 소유 turn과 중복 egress guard를 둔다

MCP는 “AI가 텔레그램으로 내보내는 도구”에 가깝고, 브릿지는 “텔레그램이 로컬 AI 터미널을 조종하는 데몬”에 가깝다. 방향이 다르다.

공식 앱과의 차이

공식 모바일 앱은 접근성이 뛰어나다. 설치가 쉽고, 클라우드 계정과 잘 붙고, 일반 대화에는 가장 편하다. 사용자가 단순히 질문하고 답을 받는다면 공식 앱이 더 낫다.

브릿지가 필요한 쪽은 다음 조건이 붙을 때다.

- 로컬 repo의 현재 git 상태를 그대로 써야 한다.
- 이미 tmux 안에서 긴 작업이 돌아가고 있다.
- 로컬 파일, 스크린샷, 생성물 경로가 작업의 핵심이다.
- 승인 프롬프트나 선택 메뉴를 폰에서 처리해야 한다.
- 터미널-origin 작업도 폰으로 상태를 받고 싶다.
- 서비스·watchdog·로컬 상태 파일까지 내 운영 방식에 맞춰 제어하고 싶다.

공식 앱은 “공식적으로 관리되는 별도 사용면”이고, 브릿지는 “내 로컬 세션의 원격 조종면”이다. 둘은 경쟁 관계라기보다 용도가 다르다.

브릿지의 장점

브릿지의 장점은 실전 운영에서 나온다.

첫째, 세션의 연속성이 좋다. 터미널에서 시작한 작업과 텔레그램에서 이어 보낸 요청이 같은 세션 안에 남는다. 작업 흐름을 나중에 찾기 쉽다.

둘째, 로컬 환경에 강하다. AI가 보고 있는 작업 디렉터리, 테스트 도구, 빌드 스크립트, 브라우저, PDF 생성 도구가 모두 내 컴퓨터 기준이다. 클라우드 세션에 파일을 다시 올리고 내리는 절차가 줄어든다.

셋째, 장시간 작업에 맞다. Codex 브릿지는 typing, progress, flow mirror, final answer mirror 같은 상태 신호를 갖고 있다. Claude 브릿지도 flow mirror를 flag로 켤 수 있다. “죽었나?”를 줄이는 장치다.

넷째, 입력 형태가 넓어진다. 사진, 이미지 문서, 동영상, 음성, 일반 파일을 로컬에 저장한 뒤 AI에게 경로로 넘길 수 있다. 폰에서 캡처한 자료를 바로 작업 세션에 넣는 흐름이 자연스럽다.

브릿지의 단점

브릿지의 단점도 명확하다.

설치가 더 복잡하다. BotFather, 토큰 파일, chat_id, tmux, 서비스 파일, watchdog, hook 같은 단어가 나온다. 공식 앱보다 손이 많이 간다.

보안 책임이 사용자에게 온다. 봇 토큰은 비밀번호처럼 다뤄야 하고, chat_id allowlist를 잘못 열면 안 된다. 텔레그램 메시지가 로컬 AI 세션 입력이 된다는 사실을 잊으면 안 된다.

운영 책임도 사용자에게 온다. 서비스가 죽었는지, polling이 멈췄는지, 세션 로그 cursor가 맞는지, 중복 답장이 생기는지 봐야 한다. Claude·Codex 브릿지의 setup wizard와 doctor가 이 부담을 줄여 주지만, 0으로 만들지는 못한다.

마지막으로 공식 지원이 아니다. 공개 repo 도구이고, 로컬 환경 차이를 완전히 제거할 수 없다. 업무상 중요한 환경에서는 작은 범위에서 먼저 검증해야 한다.

선택 기준

다음 표로 결정하면 쉽다.

상황	추천
단순 질문, 일반 대화, 클라우드 기반 작업	공식 앱
AI가 텔레그램으로 알림만 보내면 됨	MCP 도구 또는 간단한 Bot API 스크립트
내 로컬 tmux 세션을 폰에서 조종해야 함	텔레그램 브릿지
승인 프롬프트를 폰 버튼으로 처리해야 함	Codex Telegram Bridge REPL 모드
Claude Code transcript 기반으로 최종답만 받고 싶음	Claude Telegram Bridge
설치·서비스 운영이 부담스러움	공식 앱부터 시작

이 장의 결론

공식 앱과 MCP는 나쁜 선택이 아니다. 목적이 다를 뿐이다. 이 책의 목적은 “로컬에서 보이는 AI 터미널 세션을 폰에서 계속 운영하는 것”이다. 그 목적에는 브릿지가 더 직접적인 해법이다.

PART

2부. 설치하기

Claude 브릿지와 Codex 브릿지를 로컬에서 붙이는 순서

3장. 공통 준비: 봇, chat_id, tmux, 토큰 경계

각 브릿지는 설치 방식이 다르지만 준비물은 비슷하다. 텔레그램 봇 하나, 허용할 chat_id 하나, 로컬 tmux 세션 하나, 비밀 값을 저장할 설정 파일이 필요하다.

이 장에서는 실제 설치 전에 공통으로 확인할 항목을 정리한다.

준비물 목록

준비물	이유
Telegram 계정	봇과 대화할 사용자 계정
BotFather로 만든 Telegram bot	브릿지가 메시지를 읽고 답장을 보내는 Bot API 주체
bot token	브릿지가 Bot API를 호출할 때 쓰는 비밀 값
chat_id	내 계정 또는 내 대화만 허용하기 위한 숫자 id
Python 3.10 이상	두 브릿지 모두 Python 기반
tmux	보이는 Claude Code/Codex CLI 세션을 안정적으로 붙잡기 위함
Claude Code 또는 Codex CLI	실제 조종할 로컬 AI 터미널

토큰은 예시에도 실제 값을 넣지 않는다. 아래처럼 placeholder만 쓴다.

```
bot token: 로컬 설정 파일에만 저장
chat_id: 123456789
```

BotFather에서 봇 만들기

텔레그램에서 BotFather를 열고 새 봇을 만든다. BotFather는 bot token을 보여 준다. 이 token은 텔레그램 채팅방에 다시 보내지 않는다. 로컬 터미널의 설정 wizard나 로컬 파일에만 붙여 넣는다.

실수하기 쉬운 지점은 “내 봇에게 token을 보내는 것”이다. 그러면 token이 텔레그램 대화 로그에 남는다. 브릿지 설치 과정에서 텔레그램 봇에게 보내야 하는 것은 보통 /start나 /ping 정도다. token은 로컬 터미널로 간다.

chat_id는 왜 필요한가

브릿지는 기본적으로 polling 방식으로 bot update를 읽는다. 이때 아무 사용자의 메시지나 받아들이면 안 된다. 그래서 허용할 chat_id를 하나 정한다. Claude 브릿지와 Codex 브릿지의 setup wizard는 /start를 보내게 한 뒤 chat_id를 자동으로 감지한다. 수동으로 설정할 때는 CLB_CHAT_ID 같은 설정 값에 직접 넣는다.

chat_id allowlist는 가장 중요한 안전 경계다.

```
허용 chat_id: 내 개인 대화 하나  
거부 대상: 다른 사용자, 공개 그룹, 알 수 없는 update
```

처음 설치할 때는 개인 DM 하나만 허용하는 것을 권한다. 그룹방, 여러 명, topic 기반 운영은 나중 문제다.

tmux 세션 이름 정하기

브릿지는 터미널 창을 직접 조작해야 한다. 그래서 tmux 세션 이름이 중요하다.

Claude 브릿지 기본 예시는 다음 형태다.

```
tmux -L default new -s claude  
claude
```

Codex 브릿지 REPL 모드 기본 예시는 다음 형태다.

```
tmux -L codex new -s codex  
codex
```

-L은 tmux socket 이름이고, -s는 session 이름이다. 브릿지 설정의 socket/session 값과 실제 tmux 값이 맞아야 한다.

설정 파일 권한

토큰과 chat_id가 들어가는 파일은 다른 사용자가 읽지 못하게 한다.

```
chmod 600 ~/.config/telegram-agent-bridge.env  
chmod 600 ~/.config/claude-telegram-bridge/token.json
```

chmod는 해당 파일이 만들어진 뒤에 실행한다. 파일이 없다는 오류는 token 오류가 아니다. wizard로 설치하면 이 권한은 자동으로 맞춰진다.

개발용 노트북처럼 혼자 쓰는 컴퓨터에서도 이 습관을 유지한다. 나중에 백업, sync, repo commit 과정에서 비밀 파일이 섞이는 사고를 줄인다.

설치 방식

브릿지	설치 방식	장
Claude	공개 패키지(<code>pipx install</code>) + setup wizard	4장
Codex	공개 패키지(가상환경 <code>pip install</code>) + setup wizard	5장
Cursor·Grok	저장소를 <code>git clone</code> 해 직접 실행	8장

저장소를 clone할 때 예시는 홈 아래 `src`를 쓰지만, 원하는 작업 폴더를 써도 된다. 중요한 것은 설정 파일에 넣는 hook 경로, 실행 경로, service PATH가 실제 위치와 일치하는 것이다.

처음부터 서비스로 띄우지 않는다

처음 설치할 때 바로 background service부터 띄우면 어디서 막혔는지 보기 어렵다. 먼저 foreground로 실행한다.

```
claude-telegram-bridge run
codex-telegram-bridge doctor
```

`/ping`과 일반 프롬프트가 정상 동작하는 것을 본 뒤 service나 watchdog을 켜다. 원격 조종 도구는 처음부터 자동화하지 말고, 손으로 한 번 통과시킨 뒤 자동화한다.

작업 디렉터리를 좁힌다

Codex 브릿지에는 `TAB_WORKDIR` 설정이 있다. 기본값은 홈이지만, 초보자는 특정 작업 폴더로 좁히는 편이 안전하다.

```
TAB_WORKDIR=$HOME/projects/my-workspace
```

AI 세션이 접근하는 파일 범위를 줄이면 실수의 폭도 줄어든다. 특히 텔레그램은 폰에서 빠르게 입력하기 쉽기 때문에, 너무 넓은 작업 디렉터리와 강한 권한을 동시에 주지 않는 것이 좋다.

처음 쓸 엔진 하나만 고른다

처음에는 엔진 하나만 연결한다. Claude는 4장, Codex는 5장, Grok·Cursor는 8장이다. 같은 봇 token을 두 브릿지에서 쓰면 polling 충돌이 생긴다. 엔진을 늘릴 때는 엔진마다 새 봇을 만든다.

이 장의 결론

브릿지 설치의 절반은 코드가 아니라 경계 설정이다. 봇 토큰은 로컬에만 저장하고, `chat_id`는 한 명만 허용하고, tmux 이름을 맞추고, 처음에는 foreground로 검증한다. 이 네 가지가 뒤 설치를 단순하게 만든다.

4장. Claude Telegram Bridge 설치하기

확인 기준: 2026-10-18, Claude Telegram Bridge v0.14.8. 브릿지 버전과 Claude Code 버전은 서로 다르다.

Claude Telegram Bridge는 이미 실행 중인 Claude Code tmux 세션을 텔레그램에서 조종하는 데몬이다. 지금은 공개 패키지로 설치하고, 설정 wizard가 token 저장, chat_id 감지, hook 등록까지 처리한다. 이 장은 wizard를 쓰되, 서비스 등록은 뒤로 미루고 foreground에서 먼저 확인하는 순서로 진행한다.

wizard가 대신 만들어 주는 파일은 다음과 같다. 문제를 풀 때 어디를 볼지 알기 위해 이름만 기억해 둔다.

파일	역할
~/.config/claude-telegram-bridge/token.json	BotFather token 저장
~/.config/claude-telegram-bridge/token-registry.json	이 token을 어떤 브릿지가 쓰는지 기록
~/.config/claude-telegram-bridge/bridge.env	chat_id, tmux 이름 같은 로컬 설정
SessionStart hook	Claude 세션의 transcript와 tmux pane을 연결

1단계: Claude Code를 tmux에서 켜다

```
tmux -L default new -s claude
claude
```

권한 모드는 자신의 환경 정책에 맞춘다. 중요한 것은 Claude Code가 tmux 안에서 보이는 상태로 실행되어야 한다는 점이다. 브릿지는 이 pane에 텔레그램 메시지를 넣는다.

권한 확인 없이 실행하는 옵션(--dangerously-skip-permissions)을 쓰면 폰에서 승인 창 때문에 멈추는 일은 줄어든다. 대신 Claude가 묻지 않고 파일 수정과 명령 실행을 한다. 처음에는 일반 claude로 시작하고, 승인 창이 뜨면 터미널에서 직접 답한다.

이미 Claude Code를 쓰고 있었다면 현재 작업을 먼저 저장한다. 처음에는 깨끗한 테스트 세션에서 검증하는 편이 안전하다.

2단계: 브릿지를 설치한다

새 터미널 창을 연다. 이 창을 창 B라고 부른다. 1단계의 Claude 창은 창 A다.

```
pipx install "claude-telegram-bridge==0.14.8"
claude-telegram-bridge --help
```

pipx: command not found가 나오면 pipx를 한 번 설치하고 터미널을 다시 연다.

```
python3 -m pip install --user pipx
python3 -m pipx ensurepath
```

pipx는 브릿지마다 별도 환경을 만든다. Codex 브릿지도 같이 쓸 계획이라면 두 패키지를 하나의 가상환경에 함께 설치하지 않는다. 두 패키지가 같은 이름의 설정 모듈을 갖고 있어서, 한 환경에 넣으면 한쪽 명령이 다른 쪽 프로그램을 실행한다.

버전을 고정하는 이유는 이 장의 화면과 설정 이름을 맞추기 위해서다. 더 새 버전을 쓰려면 GitHub Releases의 변경 내역을 먼저 읽는다.

3단계: setup wizard를 실행한다

```
claude-telegram-bridge setup --language ko --no-service
```

--language ko는 브릿지가 폰으로 보내는 안내와 버튼을 한국어로 바꾼다. --no-service는 배경 서비스 등록을 건너뛴다. 서비스는 foreground 확인이 끝난 뒤에 켜다.

wizard는 다음 순서로 진행된다.

1. BotFather token을 **숨김 입력**으로 받는다. 화면과 셀 기록에 남지 않는다.
2. 새 봇의 개인 채팅에 /start를 보내라고 안내한다. 이 메시지로 내 chat_id를 감지한다.
3. token.json과 token-registry.json을 만든다.
4. SessionStart hook을 설치하고 ~/.claude/settings.json에 합친다. 기존 설정은 먼저 백업하고, 관계없는 설정과 기존 hook은 그대로 둔다.
5. tmux 세션이 없으면 지금 만들지 묻는다. 1단계에서 이미 만들었다면 그 세션을 쓴다.
6. 테스트 메시지를 한 통 보낸다.

hook은 Claude 세션이 새로 시작될 때 실행된다. wizard가 끝나면 창 A의 Claude 세션에서 /clear를 실행하거나 세션을 새로 시작한다.

4단계: foreground로 실행한다

창 B에서 실행하고 열어 둔다.

```
claude-telegram-bridge run
```

run은 wizard가 만든 설정 파일을 읽어 브릿지를 지금 창에서 실행한다. 끝내려면 Ctrl+C를 누른다.

텔레그램에서 봇에게 /ping을 보내고, 그다음 짧은 일반 프롬프트를 보낸다.

```
지금 작업 디렉터리와 세션 상태를 한 줄로 알려줘.
```

정상이라면 창 A의 Claude pane에 프롬프트가 들어가고, 최종 답변만 텔레그램으로 돌아온다.

상태는 언제든지 진단 명령으로 볼 수 있다.

```
claude-telegram-bridge doctor
```

5단계: 서비스로 바꾼다

foreground에서 /ping과 일반 프롬프트가 모두 통과한 뒤에만 서비스로 바꾼다. 창 B의 브릿지를 Ctrl+C로 끝내고 wizard를 다시 실행한다.

```
claude-telegram-bridge setup --language ko
```

이번에는 배경 서비스와 watchdog을 설치한다. 기존 token과 설정을 교체할지 물으면 방금 검증한 설정인지 확인하고 답한다. 설치를 되돌리려면 `claude-telegram-bridge uninstall`을 쓴다.

flow mirror 켜기

flow mirror는 작업 중 도구 사용 흐름을 텔레그램에 실시간 카드로 보여 주는 기능이다. 기본값은 꺼짐이다. 설정 파일에 `CLB_FLOW_MIRROR` 값을 따로 적지 않았다면 flag 파일로 켜고 끌 수 있다.

```
touch "$HOME/.config/claude-telegram-bridge/flow-mirror.on"
```

끄려면 지운다.

```
rm "$HOME/.config/claude-telegram-bridge/flow-mirror.on"
```

처음에는 flow mirror를 끈 상태로 최종 답장만 먼저 검증한다. 긴 작업을 맡길 때 켜다.

wizard 없이 수동으로 붙이고 싶다면

wizard가 하는 일은 모두 수동으로도 할 수 있다. token 파일과 registry를 직접 쓰고, hook 파일을 내려받아 `~/.claude/settings.json`에 등록하고, 환경 변수 세 개(`CLB_TOKEN_FILE`,

CLB_TOKEN_REGISTRY, CLB_CHAT_ID)로 실행하는 방식이다. 정확한 명령은 저장소 README의 수동 설치 절에 있다. 설정 값의 뜻은 부록 템플릿 3에 정리했다.

자주 막히는 지점

/ping이 안 오면 token과 chat_id부터 본다. `claude-telegram-bridge doctor`가 token 파일, registry, chat_id를 함께 점검한다.

프롬프트가 Claude에 들어가지 않으면 `tmux socket/session`을 본다. 실제 세션이 `tmux -L default new -s claude`로 만든 것인지, 설정의 `CLB_TMUX_SOCKET=default`, `CLB_TMUX_SESSION=claude`와 맞는지 확인한다.

/ping은 되는데 최종 답변이 오지 않으면 `SessionStart` hook을 본다. hook 등록 뒤 Claude 세션을 새로 시작했는지 확인한다. 브릿지 창의 로그가 무엇을 기다리는지 알려 준다.

중복 답장이 오면 MCP reply tool이나 다른 터미널 미러 hook이 같이 보내는지 확인한다. Claude 브릿지는 이를 줄이는 `PreToolUse` guard를 제공하지만, 환경에 기존 hook이 있으면 직접 정리해야 한다.

이 장의 결론

Claude 브릿지는 wizard가 token, registry, hook, tmux 연결을 대신 맞춰 준다. 그래도 순서는 같다. 서비스 없이 설치하고, foreground에서 /ping과 짧은 프롬프트를 통과시킨 뒤에 서비스로 바꾼다.

5장. Codex를 연결해 첫 답변 받기

확인 기준: 2026-10-18, Codex Telegram Bridge v0.9.19. 브릿지 버전과 Codex CLI 버전은 서로 다르다.

공식 Codex 설치·로그인 방법은 <https://developers.openai.com/codex/cli/> 에서 확인한다. 먼저 컴퓨터에서 codex가 실행되고 로그인에 끝나 있어야 한다. 이 책이나 텔레그램 봇이 ChatGPT 비밀번호를 받는 방식이 아니다. 이용 가능 범위와 한도는 해당 계정에서 확인한다.

1. 환경을 먼저 확인한다

Mac/Linux/WSL 터미널에서 아래 세 명령이 모두 버전을 출력해야 한다.

```
python3 --version
tmux -V
codex --version
```

Python은 3.10 이상이 필요하다. tmux가 없으면 Mac은 Homebrew의 `brew install tmux`, Ubuntu/WSL은 `sudo apt install tmux`를 쓴다. 명령 자체가 없다면 해당 도구 설치를 먼저 마치고 새 터미널을 연다. Windows PowerShell에서는 tmux 명령을 그대로 실행하지 않는다. 보이는 터미널을 연결하려면 WSL에서 이 장을 진행한다.

2. 브릿지를 별도 환경에 설치한다

이미 운영 중인 봇이 있다면 같은 token으로 새 브릿지를 띄우지 않는다. 처음 연결에는 새 봇과 별도 폴더를 쓴다. 한 봇의 polling 실행기는 하나다.

```
mkdir -p "$HOME/bridge-practice"
cd "$HOME/bridge-practice"
python3 -m venv .venv
.venv/bin/python -m pip install "codex-telegram-bridge==0.9.19"
.venv/bin/codex-telegram-bridge --help
```

패키지는 공개 배포본이고 설치 때 인터넷이 필요하다. 마지막 명령은 도움말만 표시한다.

이 가상환경에는 Codex 브릿지만 설치한다. Claude 브릿지 패키지과 같은 환경에 넣으면 두 패키지의 같은 이름 모듈이 겹쳐서 `codex-telegram-bridge` 명령이 Claude 브릿지를 실행한다. 4장의 pipx 설치의 별도 환경이므로 함께 써도 된다.

더 새 버전이 나왔더라도 처음 연결은 이 버전으로 하고, 올릴 때는 릴리스 노트를 먼저 확인한다.

3. 창 A와 창 B를 구분한다

창 A는 AI 터미널이다. 아래 명령을 실행하고 계속 열어 둔다.

```
cd "$HOME/bridge-practice"
tmux -L codex new -s codex
codex
```

`duplicate session`이 나오면 새로 만들지 말고 `tmux -L codex attach -t codex`로 기존 세션을 확인한다. 다른 작업 중인 세션이라면 그대로 두고, 새 이름을 정해 `setup`의 `--tmux-socket/--tmux-session` 값도 함께 맞춘다.

창 B는 브릿지 설정 창이다. 새 터미널을 열어 실행한다.

```
cd "$HOME/bridge-practice"
.venv/bin/codex-telegram-bridge setup --language ko --workdir "$HOME/bridge-practice"
```

1. BotFather token을 창 B의 숨김 입력에 넣는다.
2. 안내에 따라 새 봇의 개인 채팅에 `/start`를 보낸다.
3. 감지한 `chat_id`와 연결 모드 `repl`을 확인한다.
4. 기존 설정 교체 질문이 뜨면 작업 중인 설정인지 먼저 확인한다.
5. 배경 서비스 설치 여부를 고른다. 설치하면 창 B를 닫아도 브릿지가 계속 실행될 수 있다. 처음이라면 설치하지 않고 진행해도 된다.
6. 완료 메시지 뒤 아래 진단을 실행한다.

```
.venv/bin/codex-telegram-bridge doctor
```

`setup`과 `doctor`는 Telegram과 통신할 수 있다. `doctor`가 성공했다고 AI 답변 수신까지 끝난 것은 아니다. 서비스 설치를 건너뛰었다면 `wizard`가 출력한 실행 명령을 창 B에서 실행하고 열어 둔다. 설정 파일 경로를 추측해 새로 만들지 말고 `wizard`가 출력한 경로를 쓴다.

4. 파일 한 개가 폰과 컴퓨터에서 일치하는지 확인한다

처음에는 다음 문장 하나만 봇에게 보낸다.

```
현재 폴더에 bridge-check.txt를 만들고 BRIDGE_OK만 적어줘.
다른 파일은 수정하지 말고 완료한 파일 경로를 알려줘.
```

창 A에 같은 요청이 들어왔는지 본다. 승인 질문이 뜨면 자신이 요청한 파일 생성인지 확인하고 답한다. 마지막으로 창 B가 아닌 별도 터미널에서 다음을 실행한다.

```
cat "$HOME/bridge-practice/bridge-check.txt"
```

합격 기준은 세 가지다. 창 A에 원문이 1회 도착하고, 파일 내용이 BRIDGE_OK이고, 폰에 완료 답변이 도착해야 한다.

/ping만 성공했다면 연결 확인까지만 끝난 것이다. 파일은 있는데 답변이 없으면 같은 요청을 반복하지 말고 7장의 답변 미수신 경로부터 점검한다. PC가 꺼지거나 잠자기에 들어가면 응답이 중단될 수 있다.

Windows에서 시작할 때

WSL 없이 PowerShell을 쓴다면 텍스트 전용 exec 모드부터 시작할 수 있다. 먼저 `py --version`, `codex --version`이 동작하고 Codex 로그인이 끝나 있어야 한다.

```
mkdir bridge-practice
cd bridge-practice
py -m venv .venv
.venv\Scripts\python.exe -m pip install "codex-telegram-bridge==0.9.19"
.venv\Scripts\codex-telegram-bridge.exe setup --mode exec --language ko --workdir "$PWD"
.venv\Scripts\codex-telegram-bridge.exe doctor
```

exec 모드는 메시지마다 Codex를 한 번씩 실행해 답을 받는다. 이미 열린 tmux 화면을 조종하는 repl 모드와 다르다. 승인 버튼이나 진행 상황 미리처럼 화면을 보고 동작하는 기능은 repl 모드 기준이다. Windows에서 처음 쓸 때는 4절의 파일 확인 같은 작은 요청으로 먼저 동작을 확인한다.

선택 기능

작업 흐름 미리와 공개 요약(reasoning) 미리는 선택 기능이다. v0.9.19 기준 CRB_FLOW_MIRROR 기본값은 꺼짐이고, CRB_REASONING_MIRROR는 켜짐이다. 최종 답변만 받으려면 wizard가 안내한 설정 파일에 CRB_REASONING_MIRROR=0을 넣는다. 이 값은 표시 방식만 바꾼다. 계정 사용 한도를 늘리거나 다른 모델로 바꾸지 않는다. 먼저 최종 답변 수신을 확인한 뒤 필요한 기능만 켜다.

- 설치·옵션: <https://github.com/ssamssae/codex-telegram-bridge/tree/v0.9.19>
- 변경 내역: <https://github.com/ssamssae/codex-telegram-bridge/releases/tag/v0.9.19>
- Codex 공식 안내: <https://developers.openai.com/codex/cli/>

이 장의 명령과 옵션 이름은 공개 패키지 v0.9.19의 설치·도움말과 코드로 확인했다. 봇 생성, 계정 로그인, 실제 Telegram 발신은 환경마다 다르므로 4절의 세 가지 합격 기준으로 직접 확인한다.

PART

3부. 운영하기와 다른 엔진

폰 원격 운영을 오래 굴리는 습관, 그리고 Grok·Cursor 첫 연결

6장. 안전하게 운영하기

브릿지는 설치보다 운영이 중요하다. 한 번 붙이면 폰에서 AI 세션을 깨울 수 있고, 파일을 보내고, 승인 버튼을 누를 수 있다. 편리한 만큼 경계도 분명해야 한다.

첫 운영 원칙: 한 명의 chat_id

처음에는 개인 DM 하나만 허용한다. 그룹방 운영, 여러 사용자, 주제별 routing은 나중에 생각한다. 브릿지 초반 운영에서 필요한 것은 확장성이 아니라 예측 가능성이다.

좋은 시작: 내 개인 chat_id 하나
나쁜 시작: 공개 그룹, 여러 명, token 공유

텔레그램 봇은 생각보다 쉽게 전달된다. 누군가 bot username을 알 수 있고, 실수로 그룹에 초대될 수도 있다. allowlist가 없으면 이 상황이 바로 입력 경로가 된다.

token은 비밀번호다

BotFather token은 비밀번호처럼 다룬다.

- git repo에 넣지 않는다.
- 텔레그램 봇에게 보내지 않는다.
- 로그와 스크린샷에 나오지 않게 한다.
- 설정 파일 권한을 0600으로 둔다.
- 의심되면 BotFather에서 rotate한다.

책이나 팀 문서에는 항상 placeholder만 남긴다.

```
TAB_BOT_TOKEN=REPLACE_WITH_BOTFATHER_TOKEN  
CLB_CHAT_ID=123456789
```

권한 우회 옵션을 기본값으로 만들지 않는다

Codex 브릿지에는 위험한 우회 옵션이 있다.

```
TAB_CODEX_DANGEROUS_BYPASS=0
```

이 값을 1로 바꾸면 승인·샌드박스 정책을 약하게 만들 수 있다. 작업 속도는 빨라질 수 있지만, 텔레그램 입력 하나가 더 넓은 로컬 행동으로 이어질 수 있다. 초보자용 설치나 다른 사람에게 건네는 설치 문서에서는 off를 기본으로 둔다.

Claude Code도 마찬가지다. 어떤 권한 모드로 실행할지는 각자의 업무 위험도에 맞춘다. 브릿지는 권한 판단을 대신하지 않는다.

service는 마지막에 켜다

서비스 등록은 편하다. 컴퓨터를 켜면 브릿지가 살아나고, watchdog이 죽은 서비스를 다시 올린다. 그러나 디버깅 전에는 불편하다. 어디서 실패했는지 로그를 찾기 어렵다.

권장 순서는 이렇다.

1. tmux 세션을 직접 띄운다.
2. 브릿지를 foreground로 실행한다.
3. /ping을 보낸다.
4. 짧은 일반 프롬프트를 보낸다.
5. 이미지나 파일 같은 선택 기능을 하나씩 확인한다.
6. doctor 또는 health check를 통과한다.
7. 그 다음 service/watchdog을 켜다.

이 순서를 지키면 “설치가 실패한 건지, 서비스 환경 PATH가 다른 건지, token이 틀린 건지”를 분리하기 쉽다.

중복 답장을 막는다

브릿지 운영에서 흔한 사고는 중복 답장이다. MCP reply tool, terminal mirror hook, bridge daemon이 동시에 텔레그램으로 답장을 보내면 한 질문에 여러 답이 온다.

Claude 브릿지는 bridge-owned turn 동안 MCP reply를 막는 hook을 제공한다. Codex 브릿지도 final answer dedup ring과 mirror dedup을 갖고 있다. 그래도 기존 환경에 다른 텔레그램 발송 hook이 있으면 직접 정리해야 한다.

점검 질문은 간단하다.

이 질문에 답장을 보내는 경로가 몇 개인가?

답이 2개 이상이면 하나로 줄인다. 브릿지 하나가 소유하는 turn에서는 그 브릿지가 유일한 egress path가 되는 것이 좋다.

progress mirror는 소음과 정보 사이에서 조절한다

진행 상황 미러링은 유용하다. 긴 작업이 살아 있는지 알 수 있다. 하지만 너무 자세하면 텔레그램 채팅방이 도배된다.

Claude 브릿지의 flow mirror는 flag 파일로 켜다.

```
touch "$HOME/.config/claude-telegram-bridge/flow-mirror.on"
```

Codex 브릿지는 env로 조절한다. 아래는 둘 다 켜 예시다. 기본값은 flow mirror 꺼짐, reasoning mirror 켜짐이다.

```
CRB_FLOW_MIRROR=1  
CRB_REASONING_MIRROR=1
```

혼자 쓰는 초기에는 final answer 중심으로 시작하고, 장시간 작업을 많이 맡길 때 progress 계열을 켜다. 운영자는 정보가 부족해도 불안하고, 정보가 너무 많아도 피곤하다.

로컬 경로 노출을 확인한다

AI 도구는 답변에 로컬 파일 경로를 자주 쓴다. 브릿지는 media attachment 경로를 숨기고 실제 파일을 업로드하는 기능을 제공할 수 있지만, 모든 텍스트 경로를 무조건 지워 주는 보안 제품은 아니다.

공개 공유용 답변이나 고객에게 보낼 결과라면 아래를 확인한다.

- 홈 디렉터리 경로가 답변에 남아 있지 않은가.
- 비밀 파일 이름이 보이지 않는가.
- repo 내부 private branch, 내부 task id, 개인 식별자가 그대로 나오지 않는가.
- 첨부 파일 root가 너무 넓게 열려 있지 않은가.

브릿지를 개인 remote control로 쓸 때와 외부 고객에게 전달할 때의 기준은 다르다.

장애 대응 기본 순서

응답이 없을 때는 아래 순서로 본다.

1. 텔레그램 봇 token이 맞는가.
2. chat_id allowlist가 맞는가.
3. tmux session이 살아 있는가.
4. Claude Code 또는 Codex CLI가 입력 가능한 상태인가.
5. 브릿지 프로세스가 살아 있는가.

6. service PATH에서 `codex`, `python3`, `tmux`를 찾을 수 있는가.
7. `cursor/state` 파일이 현재 세션과 맞는가.
8. `approval prompt`나 `selection prompt`에서 멈춰 있지 않은가.

이 순서가 중요한 이유는 바깥에서 안쪽으로 들어가기 때문이다. `token/chat_id`가 틀렸는데 세션 로그부터 보면 시간을 낭비한다.

이 장의 결론

브릿지는 “설치하고 잊어버리는 앱”보다 “작은 운영 계층”에 가깝다. `token`, `chat_id`, 권한 모드, `service`, 중복 발송, 로컬 경로 노출을 주기적으로 확인해야 한다. 안전한 운영은 기능을 끄는 것이 아니라, 기능이 어느 경계를 지나가는지 알고 켜는 것이다.

7장. 문제 해결과 다음 단계

브릿지를 실제로 쓰다 보면 오류 메시지보다 침묵이 더 많다. 폰에서는 아무 답이 없고, 터미널에서는 뭔가 멈춰 있고, service는 살아 있는 것처럼 보인다. 그래서 문제 해결은 “어디까지 도착했나”를 단계별로 확인하는 식이어야 한다.

증상 1. /ping도 답이 없다

가장 바깥 경로부터 본다.

```
Telegram -> Bot API -> bridge polling -> chat_id allowlist -> reply
```

확인할 것:

- bot token이 현재 BotFather token과 같은가.
- token 파일 또는 env 파일 권한이 너무 열려 있어 브릿지가 거부하지 않는가.
- chat_id가 실제 내 DM id와 같은가.
- 봇에게 /start를 보냈는가.
- webhook을 쓰던 봇이라면 polling과 충돌하지 않는가.

Claude·Codex 브릿지는 doctor 명령이 token과 chat_id 설정을 함께 점검한다. 수동으로 설정했거나 Cursor·Grok 브릿지라면 chat_id 값과 token 파일 경로를 직접 본다.

증상 2. 텔레그램은 받는데 터미널에 입력이 안 들어간다

tmux 경로 문제일 가능성이 높다.

Claude:

```
CLB_TMUX_SOCKET=default  
CLB_TMUX_SESSION=claude
```

Codex:

```
CRB_TMUX_SOCKET=codex  
CRB_TMUX_SESSION=codex
```

실제 tmux 목록을 본다.

```
tmux -L default ls
tmux -L codex ls
```

session 이름이 다르면 브릿지는 엉뚱한 곳을 보거나 아무 곳에도 입력하지 못한다.

증상 3. 입력은 들어가는데 답장이 오지 않는다

세션 로그나 transcript 연결 문제를 본다.

Claude 브릿지는 SessionStart hook으로 Claude transcript와 tmux pane을 연결한다. hook을 추가한 뒤 Claude Code 세션을 새로 시작했는지 확인한다.

Codex 브릿지는 JSONL session 로그와 cursor를 본다. service 재시작 후 cursor가 현재 세션 파일과 맞지 않으면 backfill이나 tail-scan 설정이 관여할 수 있다. doctor와 bridge 로그를 본다.

증상 4. 승인 버튼에서 멈춘다

Codex 브릿지는 approval prompt를 감지해 텔레그램 버튼으로 보낸다. 버튼이 오래 지나면 stale이 될 수 있다.

```
CRB_APPROVAL_TTL_SECONDS=300
```

프롬프트가 stale이면 텔레그램에서 다시 명령을 보내기보다 터미널 상태를 먼저 본다. Codex TUI가 아직 승인 대기인지, 이미 다른 화면으로 넘어갔는지 확인한다.

증상 5. 답장이 두 번 온다

egress path가 두 개 이상일 가능성이 높다.

확인할 것:

- MCP reply tool이 켜져 있는가.
- 별도 terminal mirror hook이 같은 답을 보내는가.
- 브릿지 두 프로세스가 동시에 polling하고 있는가.
- 같은 token을 다른 consumer가 쓰고 있지 않은가.

하나의 봇 token은 하나의 consumer가 소유하게 하는 편이 안전하다. Claude 브릿지의 token registry는 이 의도를 파일로 남기는 장치다.

증상 6. service에서는 안 되는데 foreground에서는 된다

service 환경의 PATH 문제일 가능성이 높다. 셸에서는 codex나 tmux가 보이지만, systemd/launchd에서는 PATH가 다를 수 있다.

확인할 것:

- service 파일에 Python, Codex, tmux 경로가 들어 있는가.
- env 파일 경로가 service에서 읽히는가.
- working directory가 예상과 같은가.
- macOS LaunchAgent나 systemd user service가 현재 사용자 세션에서 실행되는가.

처음부터 service로 가지 말라는 이유가 여기에 있다. foreground 성공 후 service 실패라면 원인은 대개 service 환경 차이다.

운영 고도화 아이디어

기본 설치가 안정되면 다음을 고려할 수 있다.

첫째, 작업별 prefix를 붙인다. Codex 브릿지의 TAB_PREFIX나 Claude 브릿지의 emoji/node label을 활용하면 여러 세션을 구분하기 쉽다.

둘째, 작업 디렉터리를 분리한다. 같은 repo에 여러 AI 세션이 붙으면 충돌하기 쉽다. 필요한 경우 git worktree를 나누고, 각 브릿지의 working directory를 따로 둔다.

셋째, flow mirror를 상황별로 켜다. 평소에는 final answer만 받고, 장시간 작업이나 무인 운영에서는 progress를 켜다.

넷째, 정기 health check를 둔다. /ping, doctor, service status, tmux session, token owner를 한 번에 확인하는 체크리스트를 만든다.

릴리스 노트 읽는 법

브릿지는 빠르게 바뀔 수 있다. 설치 전에 release note를 본다.

각 저장소의 GitHub Releases 페이지와 저장소 안의 RELEASE_NOTES-버전.md 파일에 변경 내역이 있다. 이 책은 2026년 10월 기준 버전(Claude v0.14.8, Codex v0.9.19, Cursor v0.4.6, Grok v0.6.6)으로 명령을 확인했다. 그 사이 크게 바뀐 예로, Claude 브릿지는 처음에는 저장소를 clone해 수동으로 설정했지만 지금은 공개 패키지와 setup wizard로 설치한다.

release note를 볼 때는 “새 기능”보다 “운영 기본값이 바뀌었는가”를 먼저 본다. 예를 들어 mirror가 기본 on인지 off인지, timeout 기본값이 바뀌었는지, service 파일 이름이 바뀌었는지 확인한다.

이 장의 결론

브릿지 문제 해결은 감으로 하는 일이 아니다. Telegram, Bot API, polling, allowlist, tmux, AI session, transcript, egress 순서로 어디까지 도착했는지 보면 된다. 이 순서가 잡히면 대부분의 장애는 작은 설정 불일치로 좁혀진다.

8장. Grok·Cursor를 선택한 독자의 첫 연결

확인 기준: 2026-10-18, Cursor Telegram Bridge v0.4.6, Grok Telegram Bridge v0.6.6. 처음에는 엔진 하나와 봇 하나만 연결한다.

이 두 브릿지는 Codex wizard와 설정 키가 다르고, 패키지 대신 저장소 코드를 직접 실행한다. 해당 CLI가 컴퓨터에 설치되어 있고, 로그인까지 끝나 일반 질문에 답할 수 있어야 한다. 계정 접근과 사용량은 해당 CLI에서 확인한다. CLI가 없으면 브릿지만 설치해도 작동하지 않는다.

- Grok: <https://github.com/ssamssae/grok-telegram-bridge>
- Cursor: <https://github.com/ssamssae/cursor-telegram-bridge>

아래 명령은 Mac/Linux 터미널 기준이다. Windows에서는 WSL 안에서 실행한다. 네이티브 Windows 설치의 이 점에서 다루지 않는다.

1. 코드와 전용 폴더 준비

선택한 엔진 하나만 실행한다.

```
mkdir -p "$HOME/src"
cd "$HOME/src"
git clone https://github.com/ssamssae/cursor-telegram-bridge.git
cd cursor-telegram-bridge
git checkout v0.4.6
git rev-parse --short HEAD
```

Grok이면 위 세 곳의 `cursor`를 `grok`으로 바꾸고, 태그는 `v0.6.6`을 쓴다. 마지막 출력은 설치한 코드 버전이다. 기존 폴더가 있으면 덮어쓰지 말고 먼저 그 설치가 사용 중인지 확인한다.

2. 토큰을 화면에 남기지 않고 저장

다음 Python 코드는 숨김 입력으로 token을 받고, 봇에게 미리 보낸 `/start`에서 개인 chat_id만 골라 표시한다. 메시지 본문과 token은 출력하지 않는다. 실행 전에 새 봇의 개인 채팅에 `/start`를 한 번 보내 둔다. 다른 polling 프로세스가 같은 봇을 쓰고 있으면 먼저 그 구성을 확인한다.

```
python3 - <<'PY'
import getpass, json, os, pathlib, urllib.request
engine = 'cursor' # Grok을 골랐다면 'grok'으로 변경
if engine not in ('cursor', 'grok'):
    raise SystemExit('Choose cursor or grok')
root = pathlib.Path.home() / '.config' / (engine + '-telegram-bridge')
root.mkdir(parents=True, exist_ok=True)
target = root / 'token.json'
if target.exists():
    raise SystemExit('Existing token file: keep it and review your current setup')
token = getpass.getpass('BotFather token: ').strip()
if not token:
    raise SystemExit('Empty token')
try:
    with urllib.request.urlopen('https://api.telegram.org/bot' + token + '/getUpdates',
    timeout=15) as reply:
        data = json.load(reply)
except Exception:
    raise SystemExit('Telegram lookup failed; check the token and connection locally')
from None
ids = set()
for update in data.get('result', []):
    chat = update.get('message', {}).get('chat', {})
    if chat.get('type') == 'private':
        ids.add(chat.get('id'))
if len(ids) != 1:
    raise SystemExit('Send /start to your new private bot; exactly one private chat is
    required')
fd = os.open(target, os.O_WRONLY | os.O_CREAT | os.O_EXCL, 0o600)
with os.fdopen(fd, 'w') as out:
    json.dump({'api_key': token}, out)
print('chat_id:', ids.pop())
print('Saved:', target)
PY
```

BotFather token과 AI 로그인은 별개다. 출력된 chat_id 숫자는 다음 단계에서 예시 값 대신 넣는다.
getUpdates가 실패해도 token을 채팅에 붙여 넣어 해결하려 하지 않는다.

3. 별도 창에서 AI를 켜고 브릿지 연결

Cursor를 골랐다면 창 A에서 실행한다.

```
mkdir -p "$HOME/bridge-practice"
cd "$HOME/bridge-practice"
tmux -L default new -s cursor
cursor-agent
```

창 B는 1단계의 저장소 폴더에서 실행한다. 123456789를 앞에서 확인한 본인 숫자로 바꾼다.

```
export CUB_TOKEN_FILE="$HOME/.config/cursor-telegram-bridge/token.json"
export CUB_CHAT_ID=123456789
python3 cursor_telegram_bridge.py
```

Grok의 창 A는 `tmux -L default new -s grok` 다음 `grok`이다. Grok의 창 B는 다음과 같다.

```
export GRB_TOKEN_FILE="$HOME/.config/grok-telegram-bridge/token.json"
export GRB_CHAT_ID=123456789
python3 grok_telegram_bridge.py
```

CLI 명령이 없거나 로그인에 안 되어 있으면 여기서 멈추고 CLI 설치 안내로 돌아간다. 기존 `tmux` 세션이 있다면 새 세션을 중복으로 만들지 않는다. 창 B는 연결하는 동안 열어 둔다. 실행을 끝내려면 창 B에서 `Ctrl+C`를 누른다. 처음 연결할 때는 자동 시작 서비스를 추가하지 않는다.

4. 합격 기준

봇에 `/ping`을 보내 연결 응답을 확인한다. 그다음 5장의 `bridge-check.txt` 생성 요청을 한 번 보내고, **터미널에 원문 도착 → 파일 내용 BRIDGE_OK → 폰 완료 답변**을 각각 확인한다.

어느 단계까지 성공했는지 기록해 두면, 도움을 요청할 때 `token`이나 전체 로그를 보내지 않아도 된다. 응답이 늦다고 파일 생성 요청을 반복해서 보내지 말고 현재 터미널부터 확인한다.

이 장의 설정 이름과 실행 파일은 위 태그의 공개 코드로 확인했다. 새 봇 로그인과 실제 발신은 환경마다 다르므로 위 세 가지 합격 기준으로 직접 확인한다.

APPENDIX

부록: 설치·운영 템플릿

체크리스트와 설정 예시는 개인 값만 채워 바로 적용할 수 있게 일반화했다.

템플릿 1. 설치 전 준비 시트

아래 표를 복사해 설치 전에 채운다. 실제 token은 표에 쓰지 말고 “로컬 파일에 저장 완료”처럼만 표시한다.

항목	값 또는 상태
사용할 브릿지	Claude / Codex / Cursor / Grok
설치 기준 버전	예: Claude v0.14.8, Codex v0.9.19
작업 OS	macOS / Linux / WSL
Python 버전	3.10 이상 여부
tmux 설치	yes / no
Telegram bot 생성	yes / no
token 저장 위치	예: 로컬 config 파일, 값은 기록 금지
허용 chat_id	숫자 id, 공개 문서에는 기록 금지
Claude tmux socket/session	default / claude
Codex tmux socket/session	codex / codex
작업 디렉터리	공개 문서에는 일반 경로로만 기록
foreground /ping 통과	yes / no
일반 프롬프트 통과	yes / no
service 등록 여부	아직 / 완료

설치 중 막히면 이 시트를 먼저 본다. 빈칸이 있는 상태에서 service 문제를 보면 원인이 섞인다.

템플릿 2. 토큰 안전 체크리스트

설치 직후와 문서 공유 전 아래를 확인한다.

- BotFather token을 텔레그램 봇 채팅방에 보내지 않았다.
- token 값이 git repo에 들어가지 않았다.
- token 값이 README, 블로그, 전자책, 캡처 이미지에 나오지 않는다.
- token 파일 권한이 0600이다.
- env 파일 권한이 0600이다.
- 허용 chat_id가 개인 DM 하나로 제한되어 있다.
- 같은 token을 여러 브릿지 프로세스가 동시에 polling하지 않는다.
- token registry 또는 운영 메모에 owner가 명확하다.
- 의심되는 노출이 있으면 BotFather에서 rotate한다.

공유용 문서에는 항상 placeholder를 쓴다.

```
TAB_BOT_TOKEN=REPLACE_WITH_BOTFATHER_TOKEN
TAB_CHAT_ID=123456789
```

템플릿 3. Claude 브릿지 설정 예시

아래는 공개 문서용 일반화 예시다. setup wizard를 쓰면 이 값은 ~/.config/claude-telegram-bridge/bridge.env에 자동으로 만들어진다. 수동으로 설정하거나 값을 확인할 때 참고한다. 실제 token, chat_id, 절대 경로는 로컬에서만 채운다.

```
CLB_NODE=claude
CLB_EMOJI=
CLB_CHAT_ID=123456789

CLB_TOKEN_FILE="$HOME/.config/claude-telegram-bridge/token.json"
CLB_TOKEN_REGISTRY="$HOME/.config/claude-telegram-bridge/token-registry.json"
CLB_STATE_DIR="$HOME/.local/state/claude-telegram-bridge"

CLB_TMUX_BIN=tmux
CLB_TMUX_SOCKET=default
CLB_TMUX_SESSION=claude

CLB_START_AT_END=1
CLB_TG_CHUNK=4096
CLB_TG_POLL_TIMEOUT=2
CLB_TYPING_MAX_SECONDS=7200
CLB_ACTIVE_TURN_STALE_SECONDS=900

CLB_FLOW_MIRROR_FLAG="$HOME/.config/claude-telegram-bridge/flow-mirror.on"
```

SessionStart hook은 wizard가 ~/.claude/settings.json에 자동으로 합쳐 넣는다. 수동으로 설정할 때는 4장의 설명과 저장소 README의 수동 설치 절을 따른다.

검증 순서:

1. `tmux -L default ls`
2. `claude-telegram-bridge run`으로 foreground 실행
3. Telegram /ping
4. 짧은 일반 프롬프트
5. `claude-telegram-bridge doctor`
6. flow mirror는 마지막에 필요할 때만 켜다

템플릿 4. Codex 브릿지 설정 예시

아래는 공개 문서용 일반화 예시다. setup wizard를 쓰면 이 값은 `~/.config/telegram-agent-bridge.env`에 자동으로 만들어진다. 실제 token과 chat_id는 로컬 private env에만 저장한다.

```
TAB_BOT_TOKEN=REPLACE_WITH_BOTFATHER_TOKEN
TAB_CHAT_ID=123456789

TAB_BRIDGE_MODE=repl
TAB_AGENT=codex
TAB_AGENT_CMD=codex
TAB_PREFIX=BOT
TAB_PREFIX_LINE=0

TAB_STATE_DIR=~/.local/state/telegram-agent-bridge
TAB_WORKDIR=$HOME/projects/my-workspace
TAB_WORKDIR_LOCK=1
TAB_TIMEOUT=600
TAB_TG_CHUNK=4096
TAB_TYPING_INTERVAL=4

TAB_LOCAL_INPUT=~/.local/state/telegram-agent-bridge/input.fifo
TAB_STDIN_INPUT=0
TAB_CODEX_DANGEROUS_BYPASS=0
TAB_CODEX_EXTRA_ARGS=

CRB_TMUX_SOCKET=codex
CRB_TMUX_SESSION=codex
CRB_TMUX_SUBMIT_KEY=Tab
CRB_FLOW_MIRROR=0
CRB_REASONING_MIRROR=0
CRB_LONG_RUNNING_PROGRESS_SECONDS=0
CRB_TYPING_LIVENESS_SECONDS=10
CRB_APPROVAL_TTL_SECONDS=300
CRB_MAX_ATTACHMENT_BYTES=52428800
```

권장 확인(저장소를 clone해 실행한다면 마지막 줄 대신 `python3 bridge_setup.py doctor`):

```
tmux -L codex ls
codex-telegram-bridge doctor
```

템플릿 5. 운영 health check

브릿지가 조용하거나 불안정할 때 위에서 아래로 확인한다.

단계	확인	결과
1	Telegram bot token이 유효한가	
2	chat_id allowlist가 내 DM인가	
3	브릿지 프로세스가 살아 있는가	
4	service/watchdog 상태가 정상인가	
5	tmux socket/session이 존재하는가	
6	Claude Code 또는 Codex CLI가 입력 가능한가	
7	transcript/JSONL cursor가 현재 세션과 맞는가	
8	승인 또는 선택 프롬프트에서 멈춰 있는가	
9	중복 egress path가 켜져 있지 않은가	
10	/ping과 일반 프롬프트가 모두 통과하는가	

장애 기록 예시:

일시:
증상:
마지막 정상 동작:
변경한 설정:
실패 단계:
조치:
다음 확인:

템플릿 6. 공식 앱·MCP·브리지 선택표

필요한 일	공식 앱	MCP 도구	텔레그램 브리지
일반 질문과 답변	좋음	보통	과함
AI가 텔레그램으로 알림 보내기	보통	좋음	가능
폰에서 로컬 tmux 세션 조종	어려움	별도 입력 루프 필요	좋음
로컬 repo 상태 그대로 작업	제한적	설계에 따라 다름	좋음
승인 프롬프트 처리	제한적	어려움	Codex REPL mode에 적합
설치 난이도	낮음	중간	중간~높음
운영 책임	낮음	중간	높음
token/chat_id 관리 필요	없음	있음	있음
장시간 로컬 작업 감시	제한적	별도 구현 필요	좋음

결정 메모:

이번 목적:
로컬 파일 접근 필요 여부:
승인 프롬프트 처리 필요 여부:
설치/운영을 감당할 사람:
선택:
선택 이유:

맷는글

텔레그램 브릿지를 설치하면 AI가 갑자기 더 똑똑해지는 것은 아니다. 대신 운영 방식이 바뀐다. 터미널 앞에 없을 때도 작업을 던지고, 긴 작업 중간 상태를 보고, 승인 프롬프트를 처리하고, 결과를 받을 수 있다. 이것만으로도 혼자 일하는 사람에게는 하루의 빈틈이 줄어든다.

하지만 브릿지는 편의 기능인 동시에 권한 경로다. 텔레그램 메시지가 로컬 AI 세션의 입력이 된다. 그래서 토큰을 숨기고, chat_id를 제한하고, 서비스 상태를 점검하고, 위험한 우회 옵션은 함부로 켜지 않아야 한다. 공식 앱이나 MCP가 더 나은 상황도 있다. 짧은 대화, 클라우드 기반 작업, 별도 로컬 파일 접근이 필요 없는 요청이라면 공식 도구가 더 단순하다.

이 책의 결론은 한 문장이다. 폰을 AI 팀의 리모컨으로 쓰고 싶다면, 브릿지는 “새 채팅방”이 아니라 “보이는 터미널을 원격 조종하는 얇은 계층”으로 설계해야 한다.

저자소개

마이너스베타스튜디오는 1인 회사 운영, AI 자동화, 앱·콘텐츠 제작을 실전 중심으로 실험하고 정리한다. 이 전자책 시리즈는 실제 운영에서 겪은 문제를 작은 원칙, 설치 절차, 체크리스트로 바꾸는 것을 목표로 한다.

무료 공개판 안내

이 전자책은 kangdaejong.com 에서 무료로 공개한 판입니다. 개인 학습·업무에 자유롭게 읽고 활용하세요. 템플릿과 예시는 자신의 작업에 맞게 수정해 쓸 수 있습니다. 저작권은 강대종(마이너스베타스튜디오)에게 있습니다.