AI 바이브/페어 개발 — CLAUDE CODE/CODEX 과정

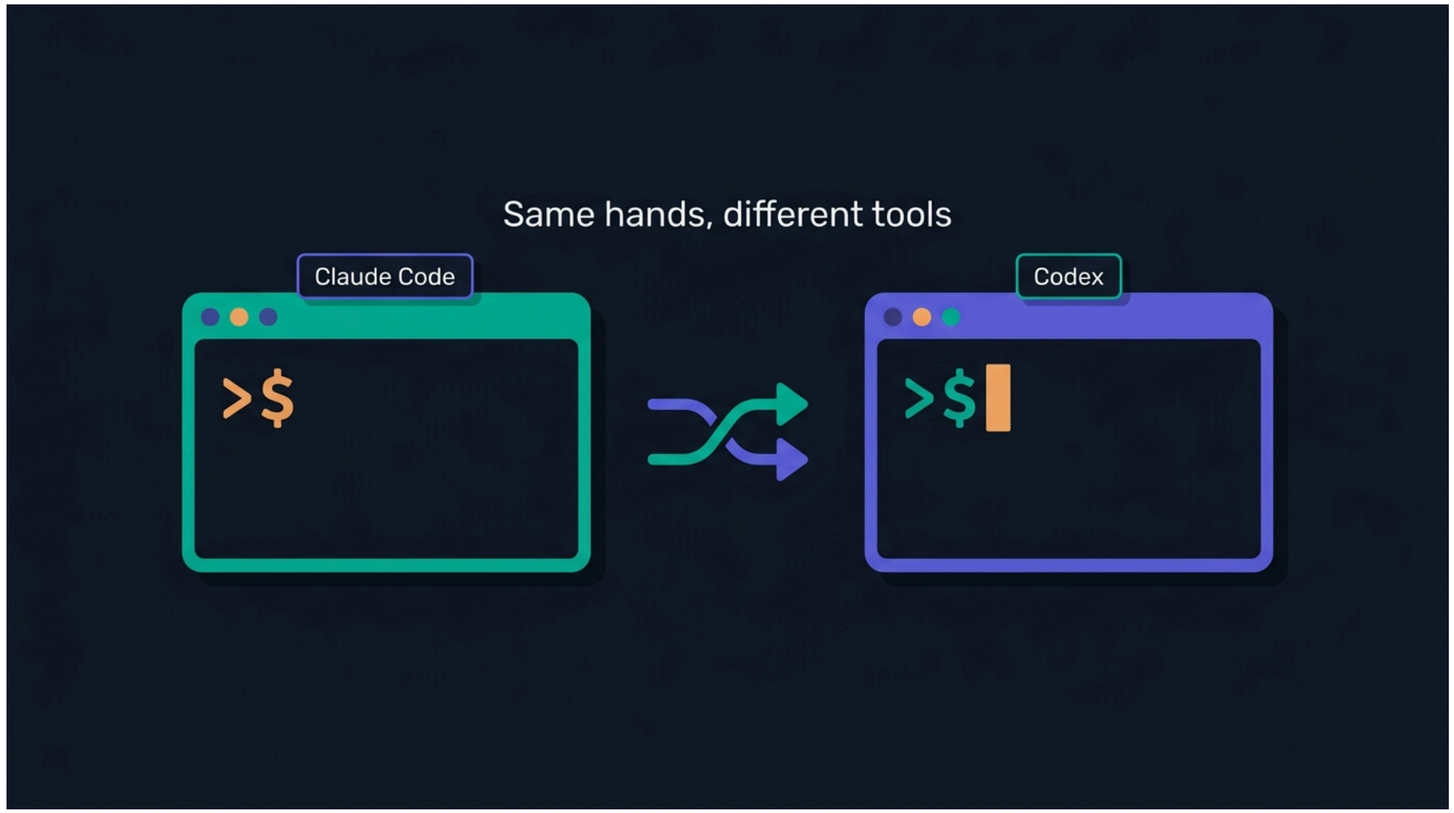
Codex 개요
승인·sandbox·AGENTS.md

Session 7 · Codex 실전 진입

두 갈래 배포 · 승인 3단계 · sandbox 3종 · AGENTS.md 3단 계층 · `codex exec` · Web UI · 실전 매핑

강사 박수현 · 🚀 젠아이랩스(GenAI Labs)

바이브코딩 2일 집중 · 프롬프트·컨텍스트·하네스 엔지니어링



Session 7 · Codex 실전 진입

PART 1

Codex 등장 배경 — *두 갈래 배포·개념 대응*

> 왜 Codex 인가 · CLI 와 Web 두 갈래 · Claude Code 와 같지 않은 세 가지.

출처: OpenAI Codex 공식 문서 AGENTS.md

Codex 두 갈래 — *CLI 와 Web UI*

Codex 는 두 갈래로 배포된다. 같은 모델·같은 AGENTS.md 규약을 공유하되 **실행 환경과 승인 UX** 가 다르다.

항목	Codex CLI	Codex Web
실행 위치	참여자 로컬 셸	OpenAI 원격 컨테이너
파일시스템	참여자 워크스페이스	격리된 클라우드 환경
인증	ChatGPT 구독 또는 API 키	ChatGPT 구독 전용
승인 UX	터미널 프롬프트	웹 UI · diff 미리보기
병렬 세션	여러 터미널·tmux	웹 UI 세션 리스트
GitHub 연동	로컬 `gh` CLI · MCP	내장 (리포·PR 자동)

📌 **핵심 결정.** 로컬 실습·짧은 반복은 **CLI**. PR 리뷰·긴 태스크·비동기 처리는 **Web**. 이 코스는 CLI 기준으로 진행하고, 파트 6 에서 Web UI 스크린을 함께 본다.

출처: OpenAI Codex 공식 문서 AGENTS.md

세 가지 차이 — *같지 않은 부분만*

대부분의 개념은 Claude Code 와 동일하고 명칭·문법만 다르다 (전체 대응표는 마지막 정리 슬라이드). **같지 않은 것은 이 셋뿐.**

🧠 모델 공급자

Codex — GPT-5.5 · GPT-5.4 · GPT-5.4-mini 계열 **Claude Code** — Opus · Sonnet · Haiku

각 모델의 성격에 맞는 도구 선택이 실무 감각.

🌐 원격 실행 UI

Codex Web — 원격 컨테이너 세션을 브라우저에서 감독.
Claude Code — 로컬 전용.

노트북 배터리·CPU 를 안 쓰는 태스크가 필요할 때 결정적 차이.

🔒 sandbox 노출도

Codex — CLI 인자·설정 파일 최상단에 sandbox 옵션 배치.
Claude Code — sandbox 지원은 있으나 하위 옵션으로 배치.

격리·안전 요구사항이 큰 팀이 Codex 를 선택하는 이유.

📌 **한 문장 요약.** "모델 계열·원격 UI·sandbox CLI 전면화" — 이 세 가지가 다르고, 나머지는 명칭·문법 차이에 그친다.

PART 2

셋업 — *설치·인증·요금·모델*

> 설치 2종 · Node 22+ · 인증 두 갈래 · 요금 6티어 · 모델 3계열 라우팅.

출처: OpenAI Codex 공식 문서 AGENTS.md

설치 경로 2종 — *npm · Homebrew*

Codex CLI 설치는 두 갈래다. Native installer 는 아직 없다. **Node.js 22+ 필수.**

npm (권장)

```
npm install -g @openai/codex
```

- **자동 업데이트** — 수동 (`npm update -g`)
- **언제** — Node.js 22+ 이미 설치, 대부분의 경우

Homebrew

```
brew install codex
```

- **자동 업데이트** — 수동 (`brew upgrade codex`)
- **언제** — macOS · brew 생태계를 이미 쓰는 경우

설치 확인

```
codex --version
codex --help
```

📌 **주의 세 가지.** - **Node 22 미만이면 설치는 되나 실행 시 오류.** `node --version` 먼저 확인. - **`sudo npm install -g` 금지.** 권한 문제로 이후 업데이트가 깨진다. nvm 또는 volta 로 관리. - **PATH 반영.** npm global bin 이 PATH 에 없으면 `command not found: codex`. shell rc 재로드.

출처: OpenAI Codex 공식 문서 AGENTS.md

인증 두 갈래 — *ChatGPT 구독 · API 키*

어느 쪽을 쓰느냐에 따라 요금 청구가 달라진다.

방식	명령	요금	언제
ChatGPT 구독	`codex login` (브라우저)	구독 요금 포함	개인 · Plus/Pro/Business
API 키	`codex login --api-key`	종량제 (토큰당)	팀 · CI · 대량 자동화

인증 성공 후 확인

```
codex
# 브라우저가 열리고 ChatGPT 로그인 → 성공 시 프롬프트가 뜸
> "이 리포 3줄로 요약해줘"
```

⚠️ **중요 세 가지.** - **ChatGPT Free · Go 는 Codex CLI 를 못 쓴다.** Plus 이상 필요. - **Codex Web 은 ChatGPT 구독 전용.** API 키로는 접근 안 됨. - **한 계정에 두 방식 병행 가능.** 로컬 실습은 구독, CI 는 API 키.

출처: OpenAI Codex 공식 문서 AGENTS.md

ChatGPT 요금 — *2026년 6월 기준*

플랜	월 요금	Codex CLI	Codex Web	대상
Free	$0	안 됨	안 됨	(Codex 미포함)
Go	$8	안 됨	안 됨	(Codex 미포함)
Plus	$20	됨	됨 (기본 한도)	개인 · 이 강의
Pro (Codex)	$100	됨 (elevated)	됨 (elevated)	매일 코드 쓰는 개발자
Pro (Max)	$200	됨 (최고)	됨 (최고)	대형 리포 · 병렬 다수
Business	$25/user	됨	됨	팀 · SSO · 감사

📌 **두 가지 주의.** - **요금·한도는 자주 바뀐다.** 슬라이드에 **"2026년 6월 기준"** 명시. - **Pro 는 두 가격이 있다.** $100 (Codex 특화) 와 $200 (Max). Codex 위주라면 $100, 전 기능 최고 한도라면 $200.

📌 **API 종량제.** Codex CLI 를 API 키로 쓰면 종량제. 모델별 토큰당 요금은 강의 직전 OpenAI Pricing 재확인.

출처: OpenAI Codex 공식 문서 AGENTS.md

모델 3계열 — *실무 라우팅*

모델 계열	성격	실무 태스크
GPT-5.5	최고 추론 · 긴 컨텍스트	어려운 리팩토링 · 아키텍처 결정 · 대형 마이그레이션
GPT-5.4 (기본)	성능·비용 균형	일반 개발 · 코드 리뷰 · 기본값
GPT-5.4-mini	저비용 · 초고속	반복 · 린트 · CI · 대량 검색

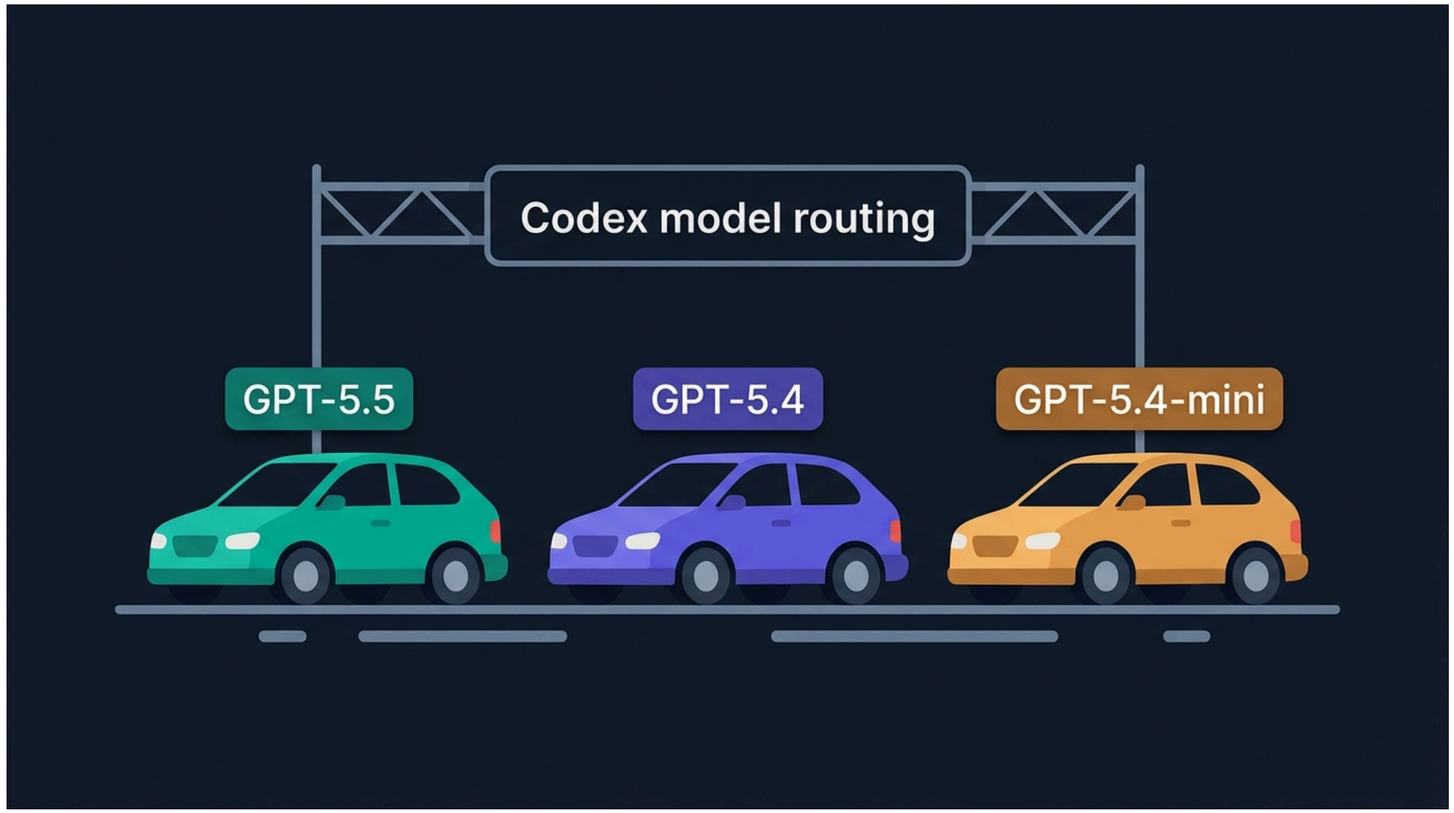
모델 전환 두 가지 방법

```
# 1. 세션 시작 시
codex --model gpt-5.5
codex --model gpt-5.4-mini

# 2. 세션 안에서 (설정에 따라)
/model gpt-5.5
```

📌 **Codex Web** 은 UI 우상단 드롭다운에서 선택. 세션 도중 전환 가능.

📌 **모델명은 계속 바뀐다.** GPT-5.5 · GPT-5.4 · GPT-5.4-mini 는 2026년 상반기 기준. 강의 직전 OpenAI Codex Models 에서 최신 라인업 재확인.

출처: OpenAI Codex 공식 문서 AGENTS.md



모델 3계열 — 실무 라우팅

PART 3

승인 모드 · sandbox — *두 축의 격자*

> 두 축 · 프리셋 3개 · 승인 3단계 · sandbox 3종 + 네트워크 스위치 · CLI 예.

출처: OpenAI Codex 공식 문서 AGENTS.md

두 축의 격자 — *승인 × sandbox*

Codex 의 실행 제어는 **두 축** 이 격자로 얽혀 있다. Claude Code 의 4개 permission mode 와 다르다.

🚦 승인 모드 (approval_policy)

Codex 가 사용자에게 **언제 승인을 묻나**.

- untrusted — 모든 편집·실행
- on-request — sandbox 밖·네트워크
- never — 절대 안 물음

3단계.

🧱 sandbox 모드

Codex 가 파일시스템·네트워크에 **어디까지 손댈 수 있나**.

- read-only — 읽기만
- workspace-write — 워크스페이스 편집
- danger-full-access — 전체 접근

3단계 (+ 네트워크 스위치).

📌 **슬로건.** "권한 = 물어보는 정책 (승인) × 손댈 수 있는 범위 (sandbox)". **이 두 축이 격자다.**

출처: OpenAI Codex 공식 문서 AGENTS.md

2 × 3 격자 · 프리셋 위치 — *한눈에*

Sandbox \ Approval	untrusted	on-request	never
read-only	🟢 **Read Only**	·	·
workspace-write	·	🟡 **Auto (기본)**	·
danger-full-access	·	·	🔴 **Full Access**

대각선 3칸이 공식 프리셋 · 나머지 6칸은 커스텀 조합.

📌 **공식 프리셋 3개.** 두 축을 매번 조합하는 대신, Codex 는 자주 쓰는 조합을 프리셋으로 묶어둔다.

📌 **감각.** Read Only = 낯선 리포 탐색 · Auto = 로컬 자기 리포 일상 · Full Access = 격리 컨테이너 배치.

출처: OpenAI Codex 공식 문서 AGENTS.md

승인 모드 3단계 — `--ask-for-approval`

정책	CLI 플래그	언제 물어보나	언제 쓰나
untrusted	`--ask-for-approval untrusted`	모든 편집·셸 실행	낯선 리포 · 안전 우선
on-request (기본)	`--ask-for-approval on-request`	sandbox 밖 액션·네트워크	일상 자기 리포
never	`--ask-for-approval never`	절대 안 물어봄	격리 환경 · CI · 배치

세션 안 승인 UX — 프롬프트가 뜨면 세 옵션.

```
[y] 이번만 허용
[a] 이 세션 내 자동 허용
[n] 거부
```

📌 **핵심 감각.** - **untrusted** — Claude Code 의 Default 모드에 대응. 매번 확인. - **on-request** — sandbox 안 편집은 자동, sandbox 밖 (예: `sudo` · 외부 네트워크) 만 승인. - **never** — **격리 컨테이너 밖에서 절대 쓰지 말 것.**

📌 **[a] 남발 주의.** 세션 내 자동 허용을 남발하면 실질적으로 승인 정책이 한 단계 내려간다. **처음에는 매번 확인.**

출처: OpenAI Codex 공식 문서 AGENTS.md

sandbox 3종 + 네트워크 스위치 — `--sandbox`

sandbox	파일 읽기	파일 쓰기	네트워크	언제
read-only	워크스페이스 안	금지	금지	탐색·검토·Plan
workspace-write (기본)	워크스페이스 안	워크스페이스 안	승인 필요	로컬 자기 리포
workspace-write + net	워크스페이스 안	워크스페이스 안	허용 (설정)	npm/pip 설치 필요
danger-full-access	전체 파일시스템	전체 파일시스템	자유	격리 컨테이너 전용

📌 **세 가지 개념.** - **workspace 의 정의.** Codex 를 시작한 디렉토리 아래. 이 밖은 못 건드림. `~/.codex/config.toml` 의 `[sandbox_workspace_write]` 섹션에 `writable_roots = ["/추가/경로"]` 로 추가 지정 가능. - **네트워크는 별도 스위치.** `workspace-write` 라도 기본은 네트워크 차단. 같은 파일 `[sandbox_workspace_write]` 아래 `network_access = true` 로 열어야 npm/pip 설치 가능. - **danger-full-access 는 진짜로 위험.** 참여자 노트북 시연 금지. 강사 격리 컨테이너에서만 한 번 스크린 캡처.

출처: OpenAI Codex 공식 문서 AGENTS.md

승인·sandbox CLI 예 — *상황별 세 조합*

```
# 탐색·검토용 — 낯선 리포
codex --sandbox read-only --ask-for-approval untrusted

# 로컬 자기 리포 (기본 프리셋 = Auto)
codex --sandbox workspace-write --ask-for-approval on-request

# 격리 컨테이너 안 배치 (위험)
codex --sandbox danger-full-access --ask-for-approval never
```

🧭 감각 잡기

- 위 세 명령만 숙지하면 나머지 조합은 프리셋으로 처리된다
- 인자 없이 `codex` 만 실행하면 Auto 프리셋으로 시작 — 일상 개발은 이 하나로 충분

⚠️ 마지막 명령 주의

`danger-full-access --ask-for-approval never` — 이 조합은 **오직 격리 컨테이너 안** 에서만.

노트북에서 그대로 치면 홈 디렉토리·시스템 파일이 위험하다.

출처: OpenAI Codex 공식 문서 AGENTS.md

로컬 CLI vs Codex Web — *언제 각각*

상황	도구	이유
짧은 로컬 편집 (5~10분)	**CLI**	셸·에디터 옆·문맥 전환 없음
긴 태스크 (30분+) · 배터리 아까움	**Web**	원격 컨테이너에서 처리
PR 리뷰 · diff 큼직하게	**Web**	웹 UI diff 뷰 편함
로컬 파일 대량 편집	**CLI**	원격 컨테이너 업로드 오버헤드
병렬 태스크 여러 개	**Web**	세션 리스트 UI 감독 편함
회사망 · 방화벽 이슈	**CLI**	Web 은 클라우드 접속 필요
노트북 없이 폰·태블릿	**Web**	브라우저만 있으면 됨

📌 **한 문장 요약.** 로컬 파일을 직접 편집하는 중이면 **CLI**, 원격에 맡겨두고 다른 일 할 때는 **Web**.

PART 4

AGENTS.md — *CLAUDE.md 대응*

> AGENTS.md 개념 · 3단 계층 · 32 KiB 상한 · 언어별 예 3개 · Claude 규약과 나란히.

출처: OpenAI Codex 공식 문서 AGENTS.md

AGENTS.md 가 뭔가 — *AI 에이전트를 위한 README*

`AGENTS.md` 는 Codex 가 리포에 진입할 때 **자동으로 읽는 규약 파일**이다. Claude Code 의 `CLAUDE.md` 와 개념이 같다.

📜 표준 오픈 포맷

2026년 상반기 Agentic AI Foundation (Linux Foundation 산하 · Cursor·Copilot·Windsurf 채택으로 사실상 표준화) 이 표준으로 채택.

지원 도구 — Codex · Cursor · Gemini CLI · Windsurf · GitHub Copilot.

✏️ 문법은 그냥 마크다운

어떤 헤딩이든 자유. Codex 는 마크다운을 통째로 파싱해 컨텍스트에 얹는다.

특별한 스키마 없음.

📦 32 KiB 상한

이 한도가 실무의 최대 함정.

큰 리포에서 root AGENTS.md 하나에 모두 몰아넣으면 한도 초과·뒤가 잘림.

📌 **언제 자동으로 읽나.** Codex 세션 시작 시. 프로젝트 루트 (git root) 부터 현재 작업 디렉토리까지 경로를 차례로 내려가며 각 디렉토리의 `AGENTS.md` 를 **모두 로드**. 하위가 상위 위에 누적.

AGENTS.md 3단 계층 — *전역·프로젝트·서브*

층	위치	성격
전역	`~/.codex/AGENTS.md`	개인이 모든 프로젝트에서 공유하는 지침
프로젝트	`<repo>/AGENTS.md`	팀 전체가 공유하는 프로젝트 규약
하위 디렉토리	`<repo>/frontend/AGENTS.md` 등	서브 패키지·모듈별 특화 규약

🧮 로드 순서·덮어쓰기

- 위→아래로 로드. 하위가 상위에 **누적** 이 원칙 (덮어쓰기 아님)
- 특정 규약을 강제로 대체하려면 `AGENTS.override.md` 를 만든다 (예: 서브 패키지에서 상위 lint 규칙을 전면 교체해야 할 때)
- 같은 디렉토리에 override 가 있으면 원본 무시

📏 32 KiB 상한 대응 · 3-tier 패턴

- `~/.codex/AGENTS.md` — 개인 표준 (1 KB 이하)
- 리포 루트 `AGENTS.md` — 프로젝트 명령·전역 관례 (3~5 KB)
- 패키지별 `AGENTS.md` — 세부 (자유)

루트 하나에 몰아넣으면 초과 → 뒤가 잘림.

⚠️ **실무 사고 다발.** 루트 AGENTS.md 에 내용을 계속 덧붙이다 32 KiB 초과, 뒤가 잘려 규약이 반영 안 되는 사고가 흔하다. **처음부터 계층으로 나눠 쓴다.**

예 1 — *Python · FastAPI*

```
# AGENTS.md — my-fastapi-app

## Setup
- Python 3.11+ (pyenv 관리)
- `pip install -r requirements.txt`
- `pip install -r requirements-dev.txt` for tests

## Commands
- Run: `uvicorn app.main:app --reload`
- Test: `pytest -x`
- Lint: `ruff check . && ruff format .`
- Type check: `mypy app/`

## Conventions
- Route 는 `app/routes/` 안에 도메인별 파일
- 응답 모델은 pydantic, 요청도 pydantic (Query/Body 강제)
- 새 엔드포인트 추가 시 `tests/routes/` 에 최소 1개 pytest 추가

## Do NOT
- `main` 브랜치 직접 커밋 금지 (feature 브랜치 → PR)
- Alembic 마이그레이션 파일 수동 편집 금지 (autogenerate 만)
```

📌 Setup · Commands · Conventions · Do NOT — **뼈대 4섹션.** 이 순서가 지침 로드 감각에 자연스럽다.

출처: OpenAI Codex 공식 문서 AGENTS.md

예 2 — *TypeScript · Next.js*

```
# AGENTS.md — my-nextjs-app

## Setup
- Node.js 22+ (nvm 관리)
- `pnpm install`

## Commands
- Dev: `pnpm dev`
- Build: `pnpm build`
- Test: `pnpm test` (Vitest)
- Lint: `pnpm lint` (ESLint + Prettier)

## Conventions
- App Router (`app/`) 사용, Pages Router 사용 금지
- Server component 를 기본, `"use client"` 는 필요 시에만
- Tailwind 유틸리티 우선, styled-components 는 legacy 만
- `components/ui/` 는 shadcn/ui 기반, 직접 편집 금지

## Do NOT
- `any` 타입 금지 (tsconfig strict)
- `.env` 파일 커밋 금지
```

📌 Next.js 팀에서 자주 나오는 함정 (`any` 타입 · Pages Router 혼용) 을 **Do NOT** 에 명시해 두면 세션마다 반복 지시가 사라진다.

출처: OpenAI Codex 공식 문서 AGENTS.md

예 3 — *Go · CLI 도구*

```
# AGENTS.md — mytool

## Setup
- Go 1.23+
- `go mod download`

## Commands
- Build: `go build -o mytool ./cmd/mytool`
- Test: `go test ./...`
- Bench: `go test -bench=. ./...`
- Lint: `golangci-lint run`

## Conventions
- `cmd/` 는 진입점만, 로직은 `internal/` 에
- Cobra 로 CLI, urfave/cli 사용 금지
- 에러는 `%w` 로 래핑, `errors.Is` · `errors.As` 로 검사

## Testing
- Table-driven tests 가 기본
- Golden file 은 `testdata/` 에
```

📌 **세 예 공통.** Setup · Commands · Conventions · Do NOT — 이 4섹션이 AGENTS.md 의 뼈대. 언어·프레임워크가 달라져도 뼈대는 그대로.

출처: OpenAI Codex 공식 문서 | AGENTS.md

CLAUDE.md 와 나란히 — *중복 유지 전략 3*

두 도구를 다 쓰는 팀에게 나오는 실무 질문 — **같은 규약을 두 파일에 중복 유지해야 하나.**

🔗 전략 A — 심볼릭 링크

```
# 리포 루트에서
ln -s CLAUDE.md AGENTS.md
```

장점. 한 파일만 유지. **단점.** 도구별 특화 지시 못 씀.

📚 전략 B — 공통 + 참조

```
<repo>/
├── AI_CONVENTIONS.md
├── CLAUDE.md    # @AI_CONVENTIONS.md
└── AGENTS.md    # @AI_CONVENTIONS.md
```

장점. 도구별 미묘한 차이 담김. **단점.** 세 파일 유지.

※ Claude Code 의 `@` 임포트와 Codex 의 참조는 문법이 미묘하게 다름 — 자세한 규격은 각 도구 문서 참조.

🎯 전략 C — 한쪽만 유지

Codex 만 → `AGENTS.md` 만. Claude 만 → `CLAUDE.md` 만.

가장 단순. 개인·소규모 팀은 이걸로 충분.

📌 **실무 권장.** 팀 단위는 **전략 B**, 개인 리포는 **전략 C** 가 표준. 처음부터 세 파일을 다 만들지 말고, **필요해질 때 쪼갠다.**

PART 5

감사 로그 · 배치 · 병렬 · MCP

> 로그 위치 · `codex exec` 배치 · 병렬 3갈래 · MCP 통합.

출처: OpenAI Codex 공식 문서 AGENTS.md

~/.codex/ — 로그·히스토리·설정

Codex 는 세션 활동을 두 곳에 남긴다. 하나는 사람용 히스토리, 다른 하나는 디버깅용 로그.

항목	위치	용도
세션 히스토리	`~/.codex/history/` · `codex resume`	이전 세션 재개·검색
TUI 로그	`~/.codex/log/codex-tui.log`	오류·크래시·업그레이드 진단
설정 파일	`~/.codex/config.toml`	승인·sandbox·MCP 서버 등 전역

세션 재개

```
codex resume              # 최근 세션 목록에서 고르기
codex resume <session-id> # 특정 세션 ID 로 재개
codex fork <session-id>   # 새 브랜치로 fork
```

📌 **로그 검색.** 문제가 재현되면 먼저 로그부터 확인한다.

```
grep -iE "error|panic" ~/.codex/log/codex-tui.log | tail -20
grep "2026-06-15T14:" ~/.codex/log/codex-tui.log
```

출처: OpenAI Codex 공식 문서 AGENTS.md

`codex exec` — *배치 · 비대화형*

Claude Code 의 `claude -p` 에 대응하는 배치 모드. **세션이 아니라 일회성 응답**을 받는다. CI · 셸 파이프 · 스크립트용.

```
# 프롬프트 한 번 · 결과만 출력
codex exec "이 리포 3줄로 요약해줘"

# @file 참조
codex exec "@README.md 한 줄 요약"

# 셸 파이프
cat production.log | codex exec "이 로그의 에러 패턴 5가지"

# JSON Lines — 스크립트 파싱용
codex exec --json "test 실패 원인 분류" > result.jsonl
```

🔧 승인·sandbox 그대로

```
# CI 안 · 격리 컨테이너
codex exec \
  --sandbox danger-full-access \
  --ask-for-approval never \
  "린트 자동 정리해줘"
```

📝 로그 남기기 옵션

`codex exec` 는 세션 로그를 남기지 않는 게 기본.

감사 필요 시 — `--output-log <path>` 로 별도 파일 기록.

📌 **자동화 감각의 시작.** `codex exec` 가 셸 파이프·alias·CI 로 확장되면 개인 도구가 된다.

출처: OpenAI Codex 공식 문서 AGENTS.md

병렬 세션 — *3갈래*

큰 리포에서는 리팩토링·테스트·문서 정리를 병렬로 굴리고 싶다. Codex 는 세 갈래로 지원.

🖥️ CLI 여러 터미널

`tmux` · `wezterm` split 으로 각 창에 `codex` 별개 세션.

서로 다른 브랜치·워크트리에서 병렬.

🌐 Codex Web 세션 리스트

Web UI 에서 태스크마다 세션을 만들면 원격에서 병렬.

노트북 리소스 안 씀.

⚙️ `codex exec` 백그라운드

셸에서 `codex exec "..." &` 로 백그라운드.

CI · 스크립트에서 자주.

git worktree 조합 — 로컬 병렬의 표준 패턴

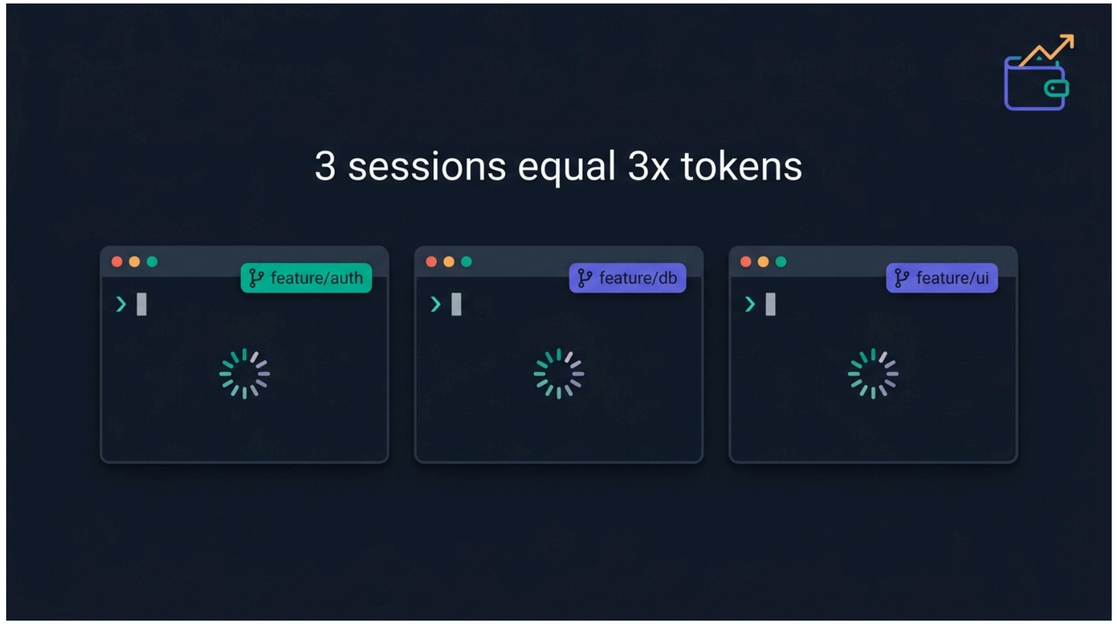
병렬 세션이 같은 파일을 동시에 만지면 편집 충돌이 난다. worktree 는 브랜치별로 작업 디렉토리를 격리해 각 세션이 자기 파일만 건드리게 한다.

```
git worktree add ../myrepo-auth feature/auth
git worktree add ../myrepo-db feature/db
git worktree add ../myrepo-ui feature/ui

cd ../myrepo-auth && codex
cd ../myrepo-db   && codex
cd ../myrepo-ui   && codex
```

⚠️ **요금 주의.** 병렬 세션은 토큰·요금도 곱해진다. Plus 는 병렬 3~4개면 한도가 빠르게 찬다. **Max 이상 권장.**

출처: OpenAI Codex 공식 문서 AGENTS.md

병렬 세션 — *그림 한 장으로 읽기*



세 터미널 · 세 브랜치 · 하나의 지갑

🖼️ 그림 읽는 법

- 세 터미널 = **auth · db · ui** 3 브랜치 병렬
- 각 스피너 = **동시에 도는 세션**
- 우상단 지갑 = **토큰·요금이 3배**

🎯 실전 감각

- worktree × CLI = **로컬 3갈래**
- Codex Web 세션 리스트 = **원격 3갈래**
- 병렬은 속도를 사는 대신 **요금을 곱한다** — Max 이상 권장

출처: OpenAI Codex 공식 문서 AGENTS.md

MCP 통합 — *두 도구가 서버 공유*

Codex 는 MCP (Model Context Protocol) 를 지원한다. Claude Code 에서 만들거나 붙였던 MCP 서버를 **그대로** Codex 에서 쓸 수 있다.

설정 위치. `~/.codex/config.toml` 에 `[mcp_servers]` 섹션.

```
# ~/.codex/config.toml

[mcp_servers.slack-summary]
command = "python"
args = ["-m", "slack_summary_bot"]

[mcp_servers.filesystem]
command = "npx"
args = ["-y", "@modelcontextprotocol/server-filesystem", "/tmp"]
```

🔌 세션 시작 시 자동 로드

Codex 는 세션 시작 시 설정된 MCP 서버를 자동으로 띄운다.

두 도구가 같은 서버 공유 — 자기가 만든 MCP 서버 하나 → Claude Code · Codex 양쪽 등록 → 둘 다 같은 도구 사용.

🔄 Codex 자체가 MCP 서버로도

`codex mcp-server` 명령으로 Codex 를 다른 도구의 MCP 서버로 노출.

예: **Claude Code 에서 Codex 를 서브 에이전트로 부르는 구조.**

📌 **MCP 가 도구 종속을 깨는 구조.** 서버 하나를 만들면 두 도구가 다 쓴다. 단, MCP 서버가 뜨는 데 시간이 걸리면 세션 시작이 느려진다 — **안 쓰는 서버는 주석 처리.**

PART 6

Web UI · 실전 매핑 · 정리

> Codex Web 세 화면 · 상황·조합 매트릭스 2 슬라이드 · Codex vs Claude · 미니 실습 3 · 정리 한 문장.

출처: OpenAI Codex 공식 문서 AGENTS.md

Codex Web UI 세 화면 — *세션·승인·Diff*

Codex Web (`chatgpt.com/codex`) 은 세 화면이 핵심.

📋 화면 1 — 세션 리스트

좌측 사이드바에 현재·과거 세션이 카드로.

각 카드에 **태스크명 · 진행 상태 · 리포·브랜치** 표시. 클릭하면 세션 상세.

💬 화면 2 — 세션 상세·승인

채팅 창 형태. Codex 가 편집·셸 실행 요청 시 인라인으로 **Approve · Reject · View Diff** 버튼.

터미널 y/n 승인보다 시각적.

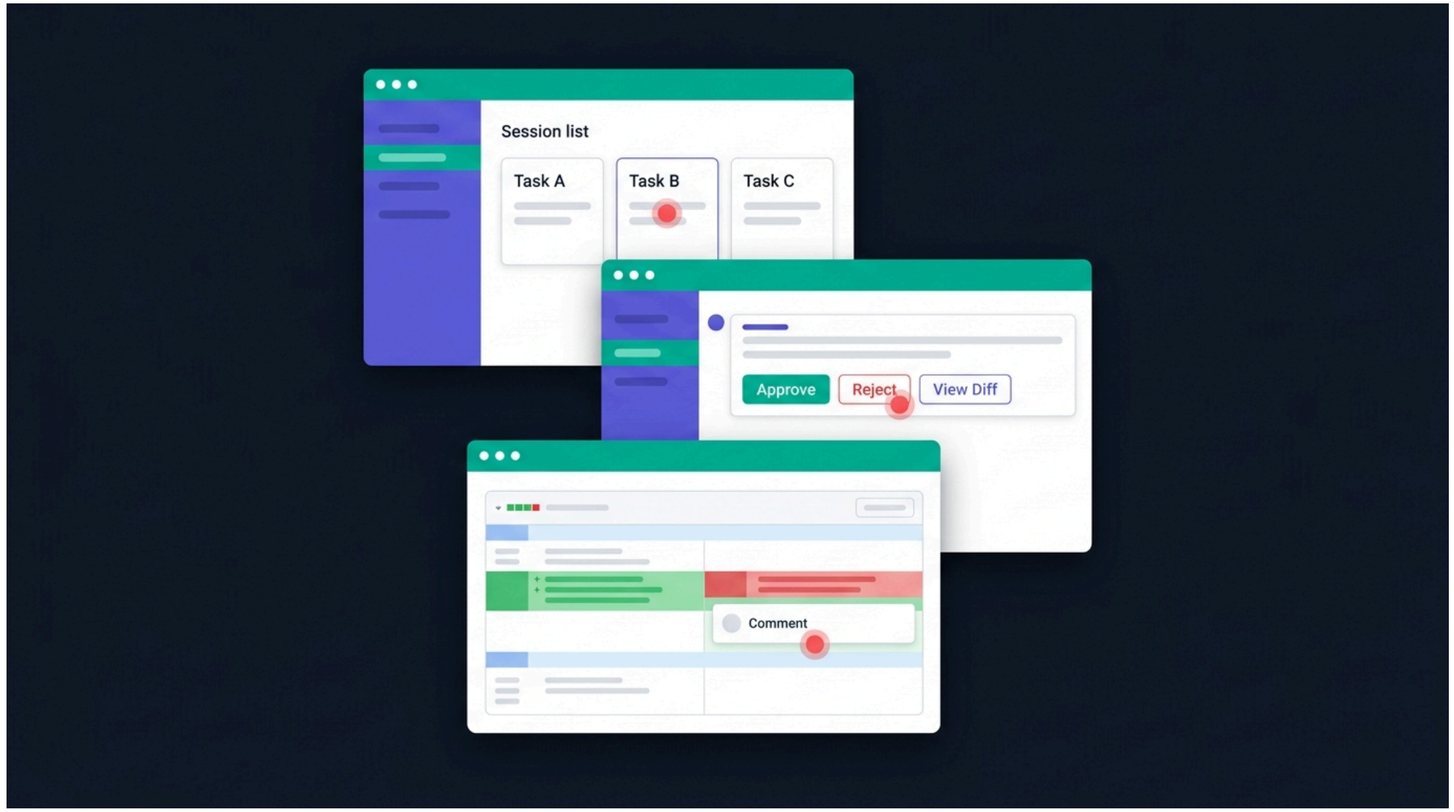
📄 화면 3 — Diff 뷰어

승인 카드 안 "View Diff" 를 누르면 GitHub 스타일 파일·라인별 diff 확장.

라인별 코멘트·거부 가능.

📌 **GitHub 연동.** Web UI 는 리포·브랜치·PR 직접 연결. **"New task" → 리포 선택 → 브랜치 선택 → 프롬프트 입력 → 원격 컨테이너 실행 → 완료 시 자동 PR 초안.**

출처: OpenAI Codex 공식 문서 AGENTS.md



Codex Web UI 세 화면 — 세션·승인·Diff

출처: OpenAI Codex 공식 문서 AGENTS.md

상황·조합 매트릭스 ① — *일상·개발*

상황	조합
낯선 리포 첫 진입 · 이해만	`codex --sandbox read-only --ask-for-approval untrusted`
로컬 자기 리포 · 일상 개발	`codex` (기본 프리셋 = Auto)
반복 lint · 정리	`codex exec --model gpt-5.4-mini "린트 오류 정리해줘"`
큰 리팩토링 시작 전	Read Only 로 계획 → Auto 로 실행
PR 리뷰 · diff 큼직이	Codex Web (`chatgpt.com/codex`)

📌 **이 5개가 일상에서 가장 자주 마주치는 상황.** 손에 익히면 승인 프롬프트에 시간을 뺏기지 않는다. **벽에 붙일 자료.**

출처: OpenAI Codex 공식 문서 AGENTS.md

상황·조합 매트릭스 ② — *자동화·격리·진단*

상황	조합
CI · GitHub Actions	`codex exec --sandbox danger-full-access --ask-for-approval never`
노트북 안 켜고 태스크 시작	Codex Web 에서 세션 생성
병렬 태스크 3개 이상	Codex Web 세션 리스트 (또는 git worktree × CLI)
로그 문제 진단	`~/.codex/log/codex-tui.log` grep
로컬 자동화 스크립트	`codex exec --json` + jq 파싱

⚠️ **CI 행의 조합은 반드시 격리 환경 안에서만 쓴다.** 개인 노트북에서 그대로 치면 홈 디렉토리·시스템 파일이 위험.

📌 이 5개는 **자동화·격리·진단**을 다룬다. 특히 `codex exec --json` + jq 조합이 팀 도입에서 효과가 가장 큰 지점.

출처: OpenAI Codex 공식 문서 AGENTS.md

Codex vs Claude Code — *팀·자동화·격리 시나리오*

시나리오	권장 도구	이유
노트북 없이 폰·태블릿	**Codex Web**	브라우저만 있으면 됨
팀 규약 · Cursor 도 쓰는 팀	**Codex** (AGENTS.md 표준)	도구 간 규약 호환
PR 자동 리뷰 · CI	**Codex Web** (GitHub 내장)	UI 감독 편함
격리 컨테이너 · sandbox 명시	**Codex**	sandbox 옵션 전면화

📌 **한 문장 요약.** 도구는 취향·팀 상황·모델 선호에 따라 갈리지만, **개념은 대응된다.** 하나를 익히면 다른 쪽은 하루면 따라간다.

출처: OpenAI Codex 공식 문서 AGENTS.md

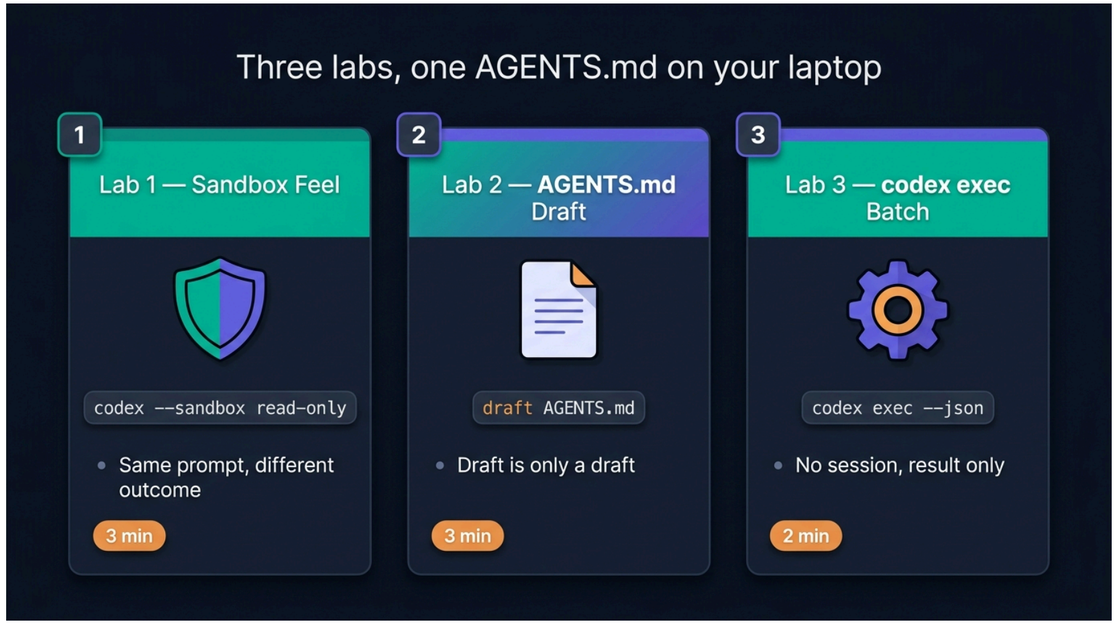
실습 1 — *첫 세션 · sandbox 관찰*

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

# 2. 세 sandbox 를 각각 시작해 감각 잡기
codex --sandbox read-only --ask-for-approval untrusted
> "이 리포 3줄로 요약해줘"
> "test.txt 만들어줘"   # 거부됨을 관찰
> Ctrl+C · Ctrl+C

codex   # 기본 = Auto (workspace-write · on-request)
> "test.txt 만들어줘"   # 승인 프롬프트 뜸
```

📌 **관찰 포인트.** sandbox 가 다르면 같은 지시의 결과가 다르다. **승인 UI 를 눈에 익히는 게 목표.**

출처: OpenAI Codex 공식 문서 AGENTS.md

실습 3개 개요



세 카드 = 세 실습 · 한 노트북에 AGENTS.md 하나

🧪 실습 흐름

1. **Lab 1** — sandbox 3종의 다른 결과 관찰
2. **Lab 2** — 자기 리포용 AGENTS.md 초안 뽑기
3. **Lab 3** — `codex exec` 로 세션 없이 배치

🎯 세션의 결정적 산출물

Lab 2 결과물 — **참여자 노트북에 AGENTS.md 초안이 남으면** 이 세션은 목표 달성.
Lab 1 은 감각 잡기, Lab 3 은 자동화 진입점.

출처: OpenAI Codex 공식 문서 AGENTS.md

실습 2 — *AGENTS.md 초안 작성*

```
# Codex 세션 안에서
> "이 리포에 맞는 AGENTS.md 초안을 만들어줘.
   Setup · Commands · Conventions · Do NOT 4개 섹션으로."

# 결과를 AGENTS.md 로 저장 (Codex 가 저장을 제안하면 승인)
```

👀 관찰 포인트

Codex 가 리포 파일을 훑고 AGENTS.md 초안을 낸다.

이 초안은 초안일 뿐 — 이후 자기 규약에 맞게 편집한다.

🎯 이 세션의 결정적 산출물

이 파일이 앞으로 Codex 세션의 뿌리가 된다.

참여자 노트북에 AGENTS.md 초안이 남으면 이 세션은 목표 달성.

📌 이미 `AGENTS.md` 가 있는 리포에서는 `AGENTS.override.md` 로 진행하거나 백업 후 실행.

출처: OpenAI Codex 공식 문서 AGENTS.md

실습 3 — `codex exec` 배치

```
# 셸에서 세션 없이 바로
codex exec "@README.md 한 줄 요약"

# JSON 출력
codex exec --json "이 리포에서 pytest 실행 방법" | head -20
```

👀 관찰 포인트

- 세션이 안 뜨고 **결과만 stdout** 으로 나온다
- 이 결과를 셸 파이프·alias·CI 에 어떻게 붙일지 상상
- **자동화 감각의 시작점**

🚀 확장 실습 (도전)

```
# 자기 셸 alias — `~/.zshrc` · `~/.bashrc` 에 등록해야 세션 넘어 유지
alias csum='codex exec "요약해줘"'

# 로그 요약
cat /tmp/app.log | csum
```

개인 도구화의 첫 성공 경험.

📌 참여자가 각자 자기 리포에서 반복. 개인 도구화의 첫 성공 경험이 이후 학습의 동력이 된다.

출처: OpenAI Codex 공식 문서 AGENTS.md

Codex vs Claude Code — *정리표*

축	Claude Code	Codex
모델	Opus · Sonnet · Haiku	GPT-5.5 · GPT-5.4 · GPT-5.4-mini
규약 파일	`CLAUDE.md` (3계층)	`AGENTS.md` (3단 + override)
실행 모드	6종 (Default · Plan · acceptEdits · auto · dontAsk · bypass)	승인 3단계 × sandbox 3종
배치 실행	`claude -p`	`codex exec`
원격 UI	(없음)	Codex Web
MCP	`claude mcp add`	`~/.codex/config.toml`
인증	Anthropic 구독 · API 키	ChatGPT 구독 · API 키
격리 실행 강조	옵션 뒤로 숨김	CLI 인자 전면화

📌 **한 장으로 스크린샷 남길 자료.** 이 표만 있으면 Codex 의 핵심 개념 대부분이 정리된다.

출처: OpenAI Codex 공식 문서 AGENTS.md

트러블슈팅 상위 3 — *참여자가 자주 부딪히는 문제*

문제	원인	조치
`command not found: codex`	npm global bin 이 PATH 에 없음	`npm bin -g` 로 경로 확인 · shell rc 재로드
브라우저 로그인 실패	회사 프록시 · SSO	Hotspot 우회 · API 키 인증 (`codex login --api-key`)
sandbox 오류로 편집 거부	read-only 모드 진입	`--sandbox workspace-write` 로 재시작 · 또는 승인 요청 확인

📌 **예상 질문 3개.** - **"ChatGPT Free 로 이 강의 따라갈 수 있나요?"** — 안 됨. Plus 이상 필요. - **"회사 프록시로 인증이 안 돼요."** — 개인 hotspot 으로 첫 인증 → 이후 API 트래픽은 프록시. 예외에 `*.openai.com` · `*.chatgpt.com` 추가. - **"AGENTS.md 를 CLAUDE.md 와 하나로 합칠 수 있나요?"** — 심볼릭 링크로 가능하나 도구별 미묘한 차이 못 씀. 팀 리포는 공통 파일 참조 (전략 B) 권장.

출처: OpenAI Codex 공식 문서 AGENTS.md

마무리 — *한 문장으로*

이 세션이 끝났다. 노트북에서 Codex 가 뜨고, 승인 모드 3단계와 sandbox 3종을 상황별로 매핑할 수 있으며, 자기 리포에 `AGENTS.md` 초안이 놓였다.

🎯 이 세션의 축 한 줄

"승인 정책 × sandbox 가 격자, AGENTS.md 가 계층 규약, `codex exec` 가 자동화 진입점."

상황이 애매하면 **승인·sandbox 어느 축에서 결정할지**, **AGENTS.md 로 이미 답이 있는지**부터 짚는다.

🚦 승인·sandbox

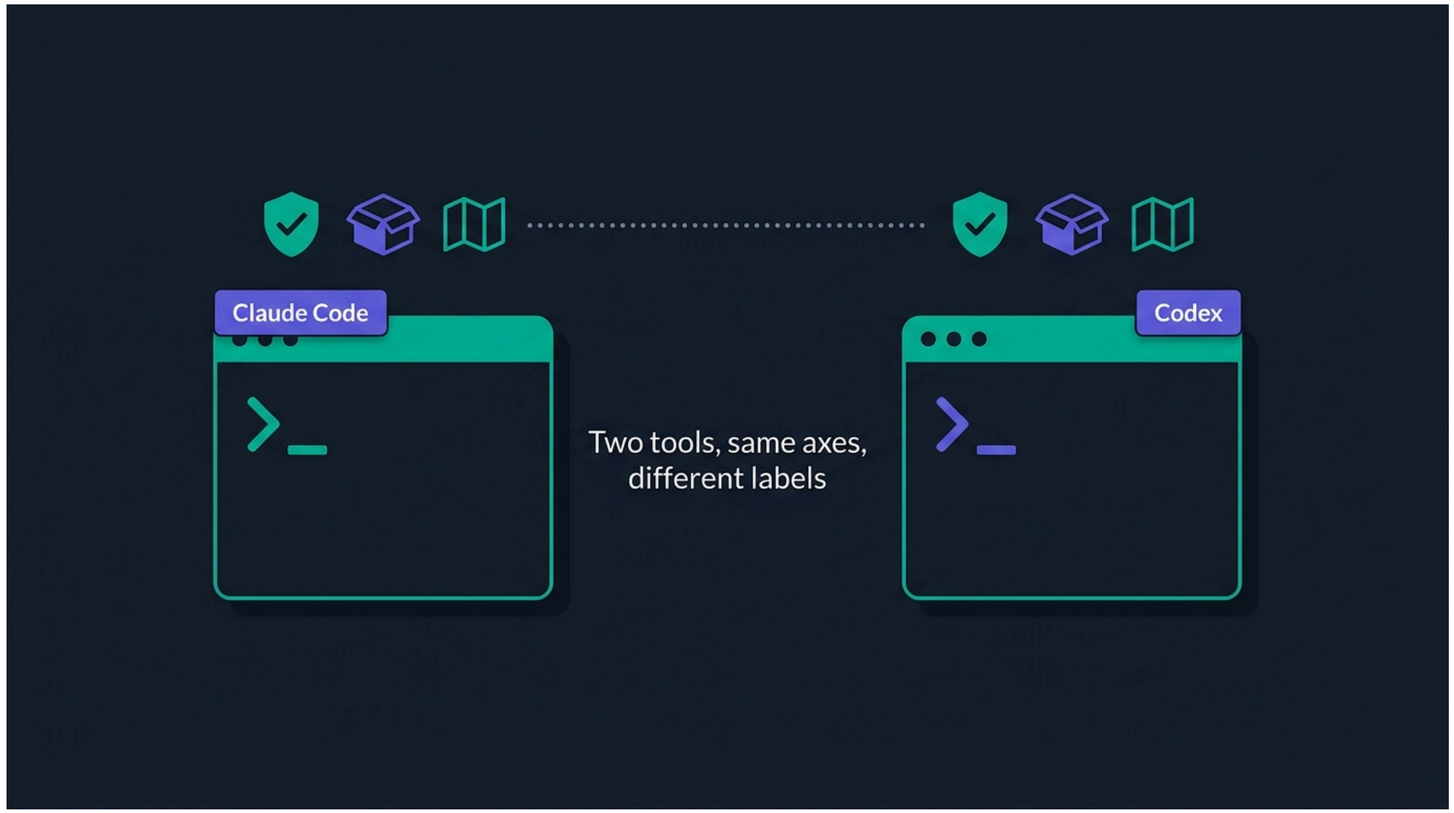
Read Only · Auto · Full Access — 프리셋 3개만 손에 익히면 격자는 이미 정리된다.

🗺️ AGENTS.md

3단 계층 · 32 KiB · override — 팀 규약이 세션마다 반복 지시를 지운다.

⚙️ codex exec

셸 파이프·alias·CI — 자동화 진입점. 개인 도구가 여기서 시작된다.



마무리 — 두 도구, 같은 축, 다른 라벨