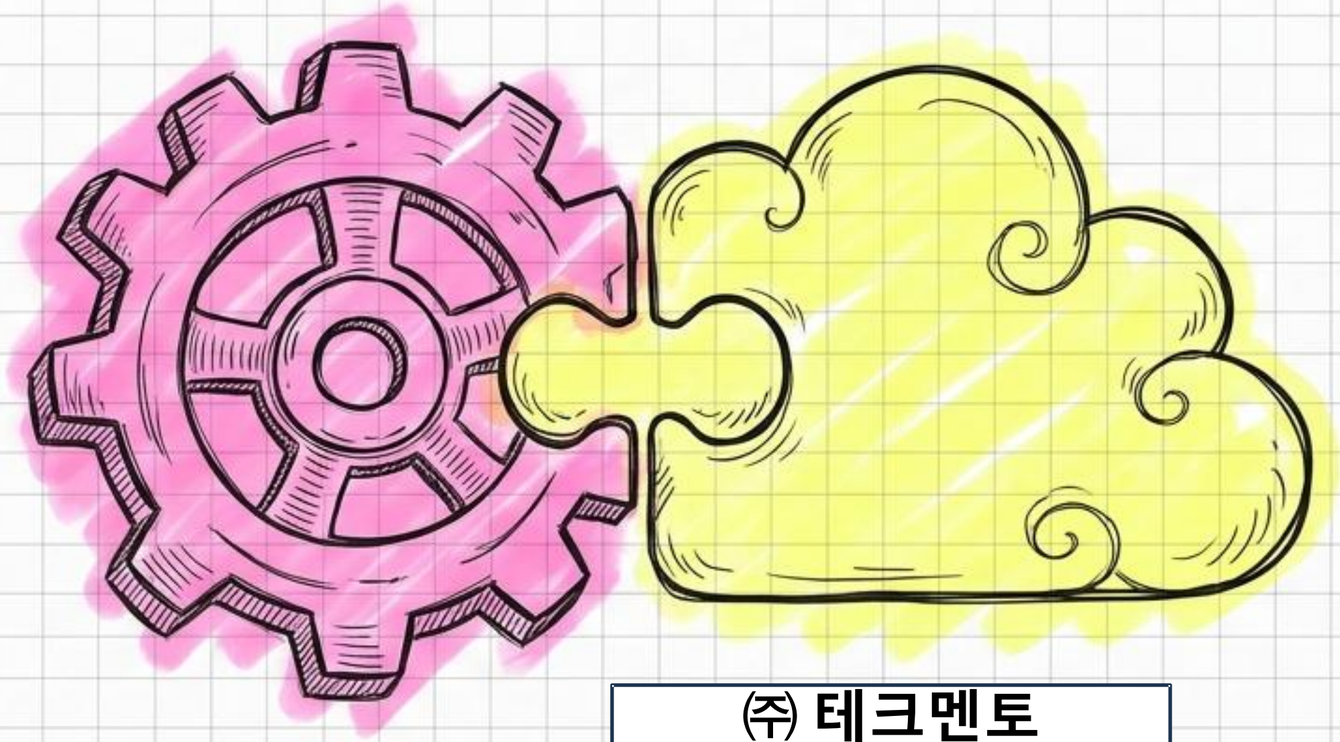
# Power Apps Code Apps

Low-Code의 세상에 Pro-Code 엔진을 얹다



㈜ 테크멘토
구병국

# 후원사

## 다이아몬드

Microsoft

## 플래티넘

Korea Software Service

(주)한국소프트웨어서비스

## 골드

go deploy

STK

TRAINOCATE

## 실버

OPEN SG Solution Group

## 브론즈

한빛미디어

# About Me

- Personal Information
  - Name: 구병국(Byeongguk Ku)
  - Position: ㈜ 테크멘토 대표
- Personal History
  - 30년간 교직(1995~2025, 고등학교, 대학교)
  - MCT(2000~현재)
  - MVP(2003~2009(Exchange Server), 2014(Office 365))
- Contacts
  - https://techmentor.kr
  - ceo@techmentor.kr

업무 생산성 / 오피스

# 따라하면서 배우는 Power Apps

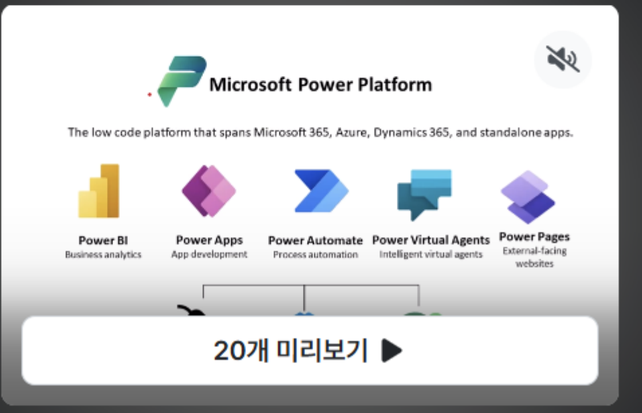
엑셀의 함수 정도만 알고 있으면 Power Apps를 활용하여 누구나 손쉽게 앱을 만들어서 현업에 사용할 수 있는 방법을 설명합니다. Power App의 개념을 이해하고 다양한 컨트롤과 꼭 필요한 함수를 ...

더보기

★★★★★ (4.8) 수강평 6개 수강생 150명

Techmentor

# microsoft365r MS Project m365 Power Apps Power Automate






kt R 리디 에 관심있는 사람들도 듣는 중!

lagoon 수강평 1 · 평균 평점 5.0

★★★★★ 5.0 60% 수강 후 작성

설명이 자세해요

무니 수강평 1 · 평균 평점 5.0

★★★★★ 5.0 40% 수강 후 작성

알기쉽게 설명해주시고 핸즈온강의라 따라하면서 빠르게 실전연습을 할수있어서 좋아요!

신규가입 25% 할인

₩57,750

25% ~~₩77,000~~

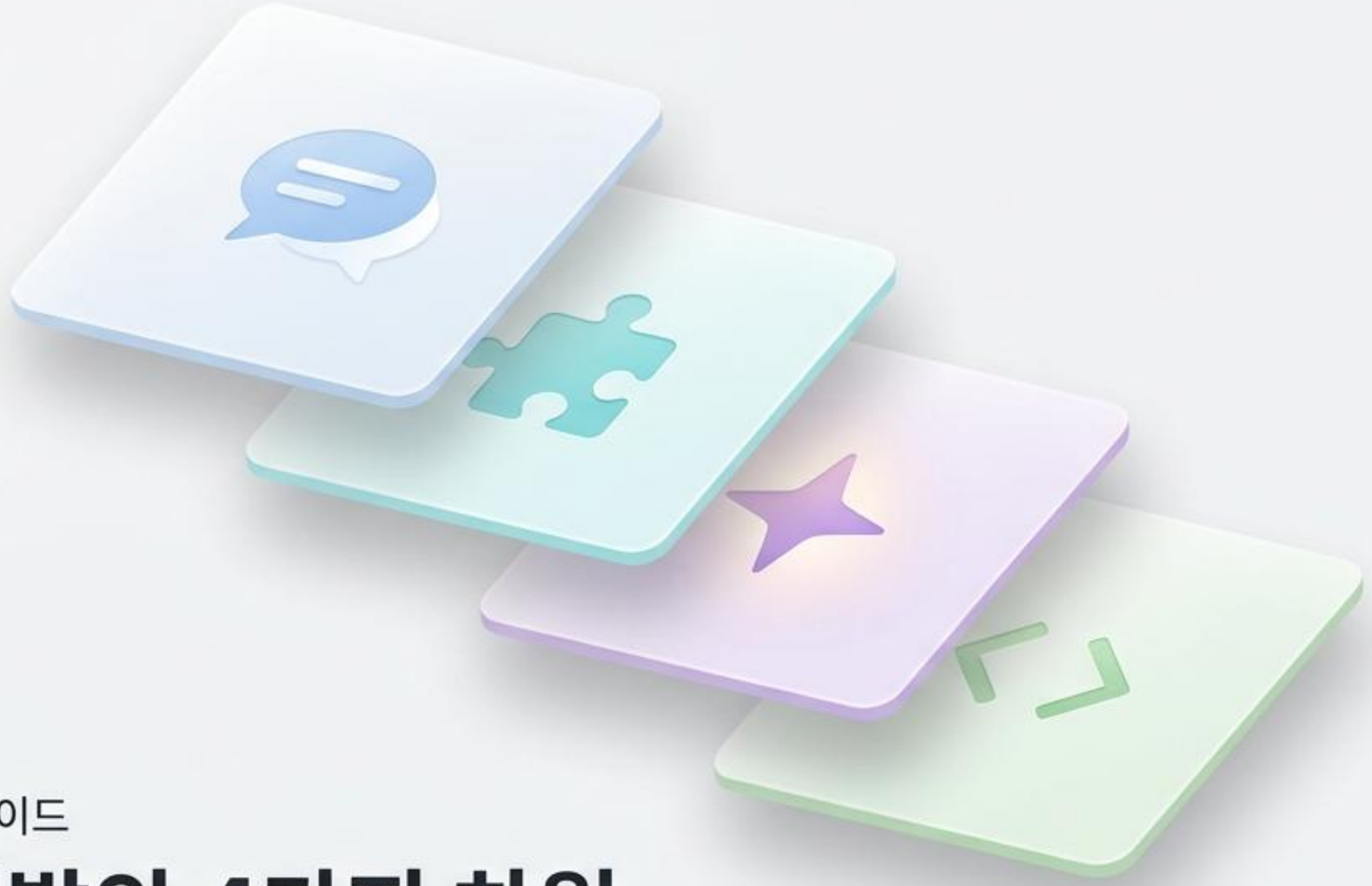
수강신청 하기

바구니에 담기

선물 조르기 / 보내기

문의하기

지식공유자 Techmentor



Power Platform 에코시스템 분석 가이드

# Power Apps 개발의 4가지 차원

No-Code부터 Pro-Code까지, 조직의 요구사항에 맞춘 최적의 개발 방식 비교

# 아이디어에서 엔터프라이즈 시스템까지

Power Apps는 단순한 업무 자동화부터 전문적인 풀스택 시스템 구축까지, 타겟 사용자와 목적에 맞춘 4가지 명확한 개발 스펙트럼을 제공합니다.

**No-Code:**
자연어 대화형 빌드
(정보 근로자)

**Low-Code:**
스튜디오 기반 시각적 빌드
(앱 메이커)

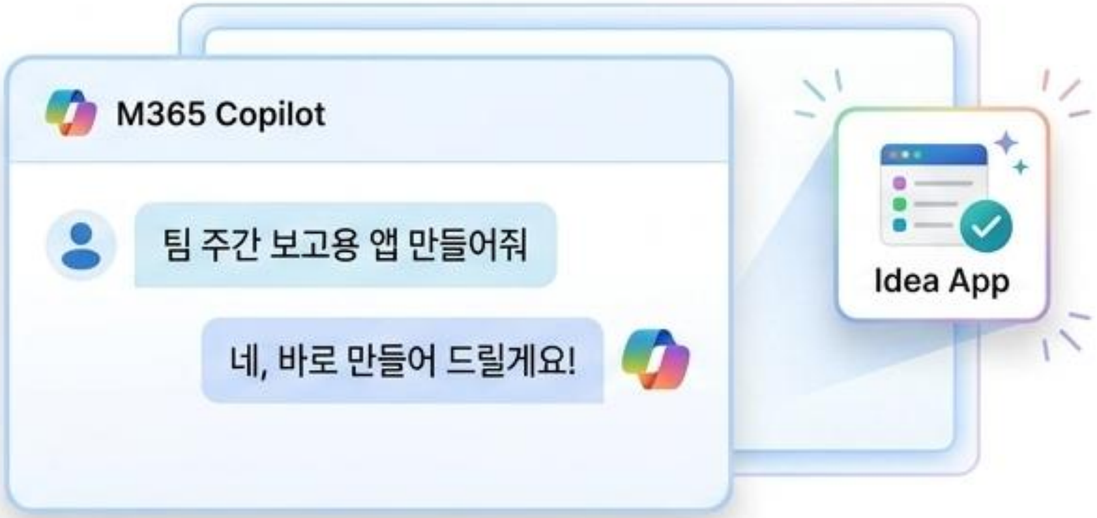
**Vibe Code:**
AI 풀스택 자동 생성
(메이커 & 개발자)

**Pro-Code:**
완벽한 통제와 코딩
(전문 개발자)

# 1. No-Code: 대화하듯 만드는 가장 빠른 앱

M365 Copilot의 'App Builder Agent'를 활용하여 일상적인 자연어로 팀 단위의 생산성 앱을 구축합니다.

- **대상:** 코딩 지식이 없는 일반 정보 근로자
- **환경:** M365 Copilot 내부에 직접 탑재 (별도 스튜디오 불필요)
- **데이터:** 앱 생성 시 자동 구축되는 전용 SharePoint 사이트

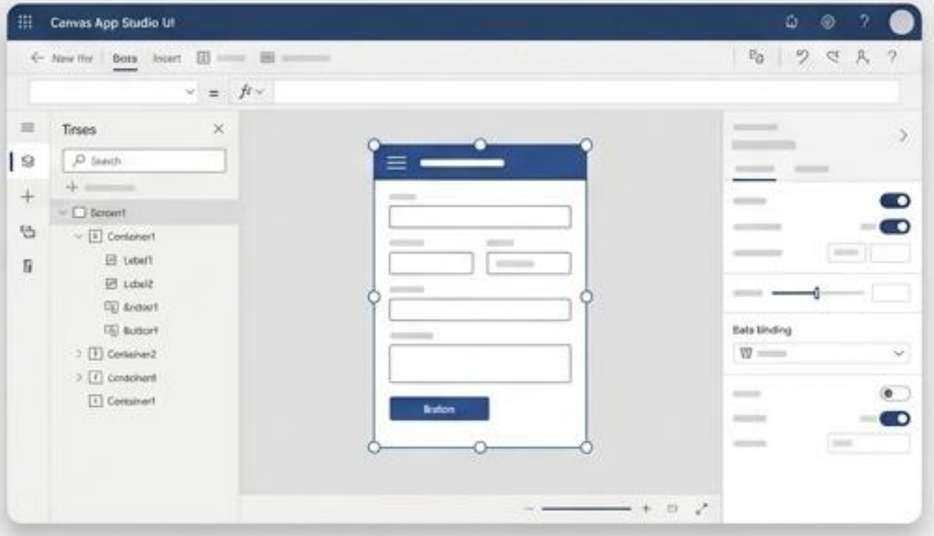



| 장점 | 단점 |
| --- | --- |
| 코딩 제로, 채팅 지시만으로 즉시 완성, 쉬운 공유 | 세부 픽셀/UI 조정 불가, 생성자 및 사용자 모두 M365 Copilot 라이선스 필수 |

# 2. Low-Code: 엔터프라이즈급 앱을 위한 스튜디오

make.powerapps.com 환경에서 Power Fx 구문과 Plan Designer를 통해 전사적 규모의 앱을 시각적으로 설계합니다.

- **대상**: 앱 메이커 (Makers) 및 비즈니스 개발자
- **환경**: 전용 스튜디오 (Canvas / Model-driven apps)
- **데이터**: Dataverse, SharePoint, SQL 등 모든 커넥터 활용 가능




## 장점

방대한 커넥터 생태계 연동, 완벽한 ALM 지원, 세밀한 UI 제어 가능

## 단점

Power Fx 문법 및 데이터 모델링 등 로우코드 도구에 대한 학습 곡선 존재

# App Builder vs. Power Apps Studio

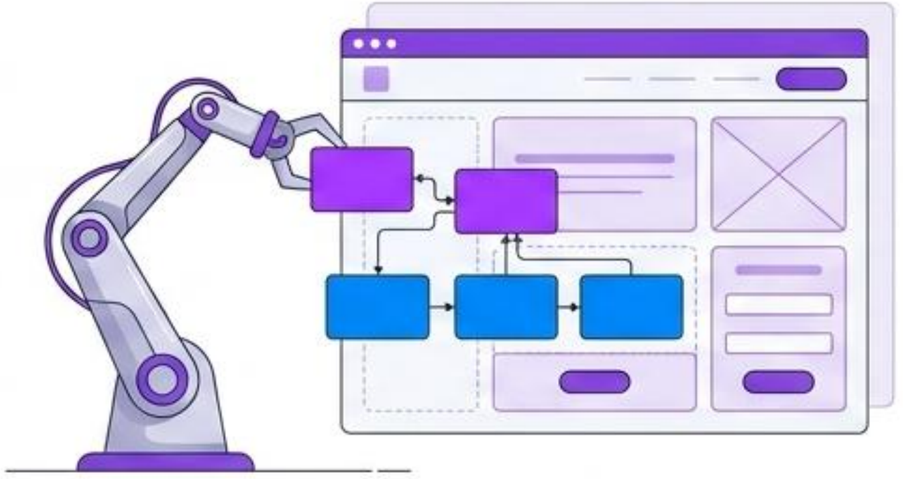
| | App Builder | Studio |
|---|---|---|
| **주요 목적** | 개인/팀 생산성 향상 | 전사적(Enterprise) 솔루션 구축 |
| **개발 환경** | M365 Copilot 내부 직접 탑재 | make.powerapps.com 전용 스튜디오 |
| **데이터 백엔드** | 전용 SharePoint 사이트 (자동 생성) | Dataverse, SQL 및 전체 커넥터 생태계 |
| **UI 제어권** | 없음 (에이전트를 통한 수정만 가능) | 완벽한 제어 (Full control) |
| **필요 라이선스** | 메이커/사용자 모두 Copilot 라이선스 필요 | 일반 Power Apps 라이선스 |

💡 시사점: App Builder는 팀을 위한 똑똑한 '비서'이며,
Studio는 전사적 통제가 가능한 본격적인 '제작 도구'입니다.

# 3. Vibe Code: AI가 생성하는 풀스택 React 앱

4개의 AI 에이전트가 협력하여 자연어 지시만으로 몇 분 만에 완벽한 React 기반의 웹 프론트엔드를 구축합니다.

- **대상:** 앱 메이커 및 전문 개발자
- **환경:** 새로운 엔드포인트 (vibe.powerapps.com) / Gen Pages
- **특징:** 10배 빠른 개발 속도, 생성된 소스 코드 전체 접근 및 직접 수정 가능



### 장점

전문 개발팀 수준의 결과물 쾌속 생성, 생성된 코드의 완벽한 접근 및 수정 권한 보장

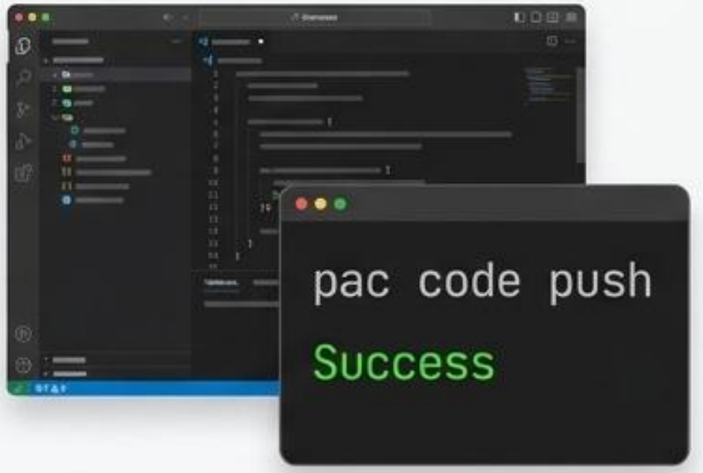
### 단점

기본 데이터 소스가 Dataverse로 제한됨, 직접 고도화 시 결국 프론트엔드 지식 필요

# 4. Pro-Code: 플랫폼의 한계를 넘는 자유

전문 개발자가 친숙한 로컬 IDE를 사용하여 플랫폼의 제약 없이 나만의 코드와 컨트롤을 완벽하게 작성하는 Code Apps 영역입니다.

- **대상:** 전문 웹 개발자 (Pro Devs)
- **환경:** Visual Studio Code 등 로컬 IDE
- **배포:** 단일 명령어(pac code push)로 Power Platform 환경에 즉시 배포




## 장점

무한한 유연성, 로컬 환경 실시간 테스트, Power Platform 커넥터 생태계 완벽 통합

## 단점

전문적인 React 및 웹 개발, CLI 도구 활용 역량 필수

# Pro-Code 방식은 언제 필요한가?

기존 로우코드 환경의 기술적 제약을 넘어, 복잡한 엔터프라이즈 요구사항을 충족하기 위한 4가지 전략적 당위성.

### 완전한 제어권 (Full Control)

가능한 영역 그 이상을 위한 맞춤형 UI/UX와 독창적 비즈니스 로직의 완벽한 구현.

### 익숙한 IDE 활용

Visual Studio Code 등 친숙한 로컬 개발 환경 유지로 개발팀의 생산성과 만족도 극대화.

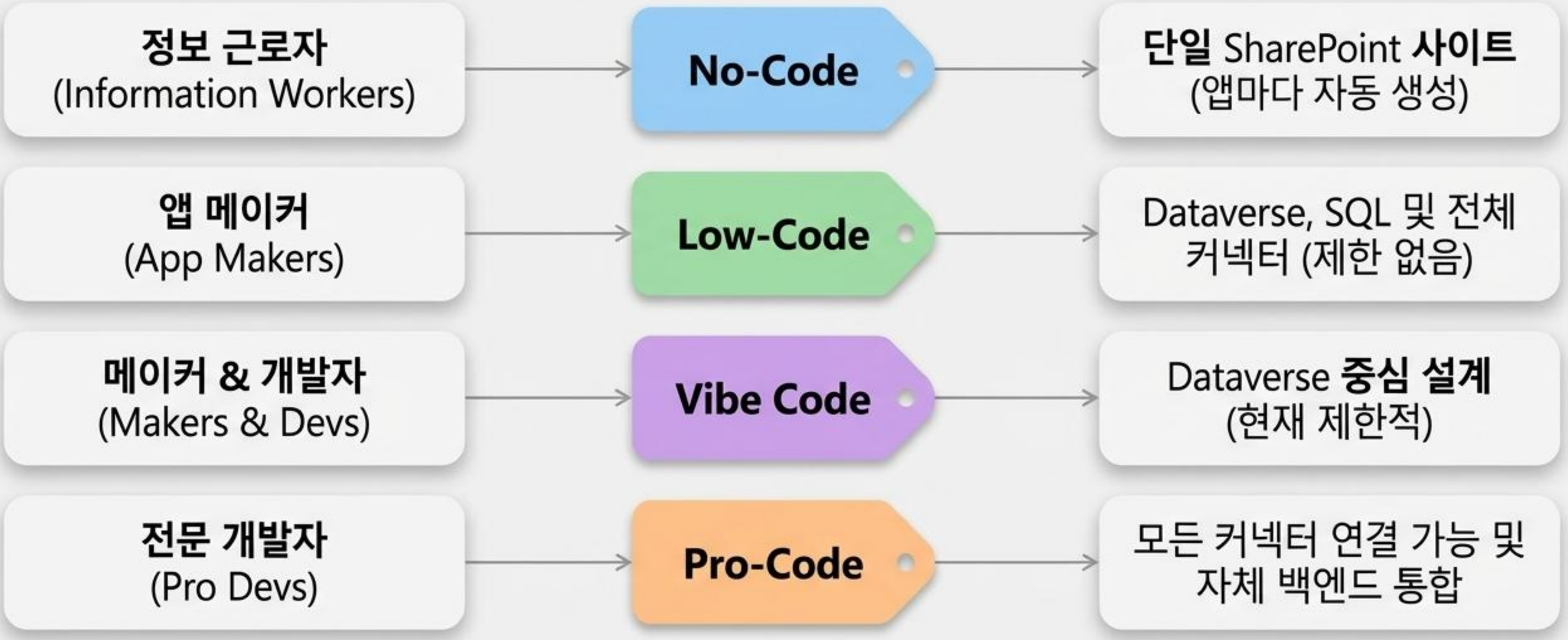
### 생태계 시너지

무한한 커스텀 프론트엔드 코드 내에서 Power Platform의 방대한 커넥터 생태계를 그대로 호출하고 활용.

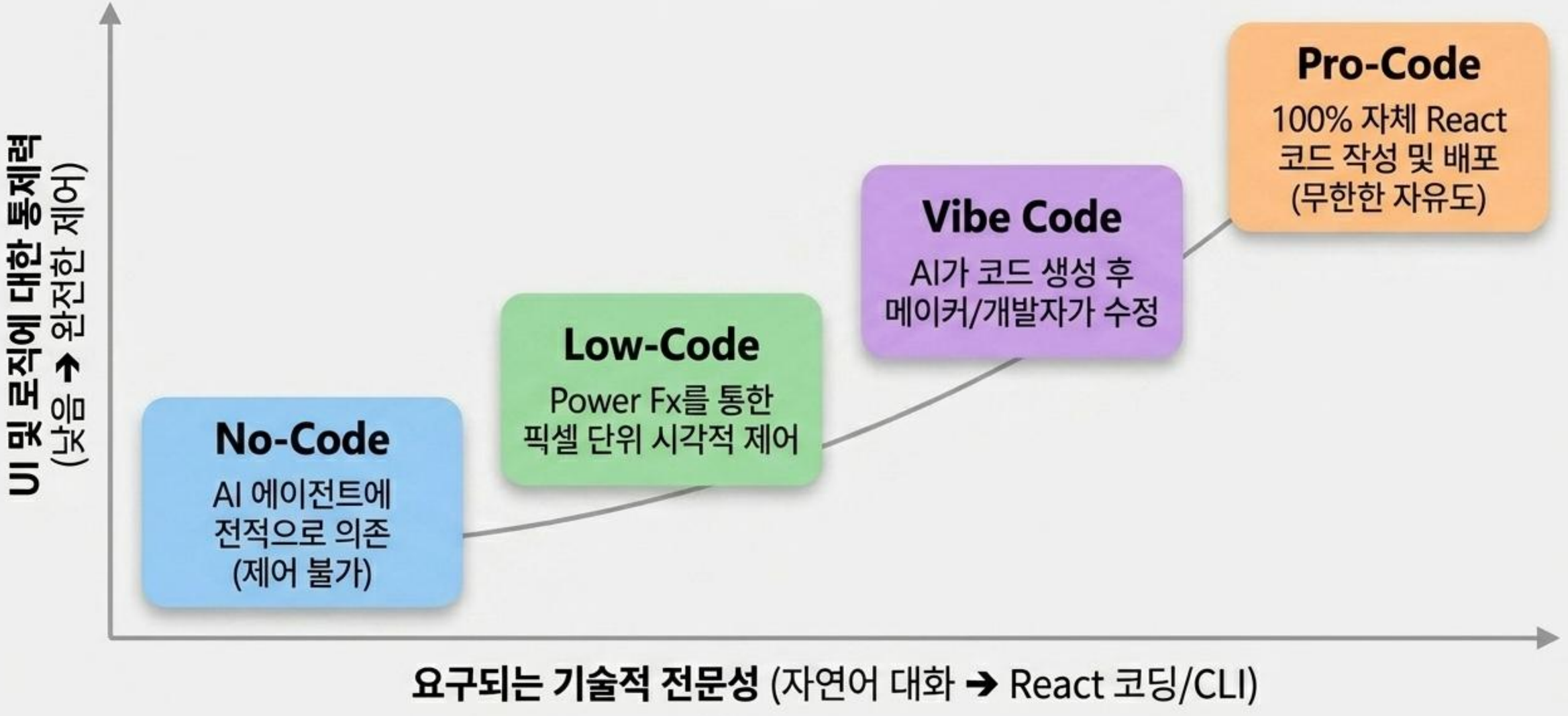
### 효율적인 배포 파이프라인

로컬 환경에서 실시간 검증 후 단일 명령어 하나로 연결된 환경에 즉각 배포 완료.

# 누가, 어떤 데이터를 다루는가?



# 통제력(Control)과 개발 전문성의 트레이드오프



# 개발자 중심의 새로운 패러다임

### 풀스택 코딩의 자유

React, TypeScript 등 친숙한 프레임워크로 완벽한 UI/UX 커스텀 제어 및 단위 테스트(Jest 등) 구현.

### 엔터프라이즈급 거버넌스

별도 인프라 구축 없이 Power Platform의 보안, Entra ID 인증, Dataverse 무결성을 그대로 상속.

### AI 보조 개발 (Vibe Coding)

GitHub Copilot 등을 활용하여 자연어 요구사항을 즉시 React 프론트엔드 코드 및 데이터 모델로 자동 생성.

# 왜 Code Apps를 선택해야 할까요?

| 구분 | Canvas Apps | Code Apps |
| --- | --- | --- |
| 대상 | 시민 개발자 (Citizen) | 전문 개발자 (Pro-Code) |
| UI 도구 | 드래그 앤 드롭 시각 편집기 | VS Code, React, Fluent UI |
| 언어 | Power Fx | TypeScript, JavaScript |
| 테스트 | 제한적 기능 테스트 | 단위 및 통합 테스트 (표준 프레임워크) |
| 활용처 | 빠른 프로토타이핑 및 단순 로직 | 복잡한 비즈니스 로직 및 외부 API 연동 |

**Code Apps는 캔버스 앱을 대체하는 것이 아니라, 기존 도구로 한계에 부딪혔던 복잡한 비즈니스 요구사항을 해결하는 프로 코드 확장판입니다.**

# [유의점] 최종 사용자를 위한 프리미엄 라이선스 요구

내 컴퓨터에서 개발하고 배포하는 것은 자유롭지만,
배포된 앱을 실행하는 모든 직원은 프리미엄 라이선스가 필요합니다.

제작 및 서버 배포 (프리미엄 불필요)

배포 완료


실제 앱 실행 환경 (프리미엄 필수)

# 3계층 아키텍처 개요

## 개발 계층

로컬 프론트엔드
(TypeScript, React 코드)


로컬 프론트엔드
(TypeScript, React 코드)

## 브릿지 계층

트릿지어 엔진

Power Apps SDK & CLI
(데이터 모델 및 서비스 변환,
power.config.json)

## 런타임 계층

Power Platform Host
(인증, Dataverse,
커넥터 호스팅)

Power Platform Host
(인증, Dataverse, 커넥터
호스팅)

# 1. 개발 계층 (Web App Project)


- **완벽한 제어권**

  캔버스 앱의 시각적 편집기를 벗어나, VS Code와 Node.js 기반의 표준 웹 개발 환경을 온전히 사용합니다.

- **모던 프레임워크**

  React 등 선호하는 JS 프레임워크와 Fluent UI 컴포넌트로 제약 없는 픽셀 퍼펙트(Pixel-Perfect) UI를 구현합니다.

- **독립적 로컬 테스트**

  단일 페이지 애플리케이션(SPA)으로 컴파일된 코드는 npm run dev 명령어를 통해 로컬 브라우저에서 즉시 실행 및 디버깅이 가능합니다.

# 2. 브릿지 계층 (SDK & CLI)

## 로컬 코드와 Power Platform을 연결하는 지능형 번역기


### Power Apps CLI

npx power-apps 명령어를 통해 프로젝트를 초기화하고 Dataverse API나 흐름(Flow)을 앱에 추가합니다.

### SDK 자동 생성

커넥터를 추가하면 TypeScript 기반의 모델과 서비스 파일이 즉시 자동 생성되어 완벽한 타입 안정성(Type Safety)을 보장합니다.

### Configuration

power.config.json을 통해 환경 변수, 데이터 소스, 연결 메타데이터를 통합 관리합니다.

# 3. 런타임 계층 (Platform Services)

## 엔터프라이즈 보안 (Entra ID)

별도의 로그인 페이지나 토큰 관리 개발 없이 SSO, MFA, 역할 기반 접근 제어(RBAC)가 자동으로 상속됩니다.

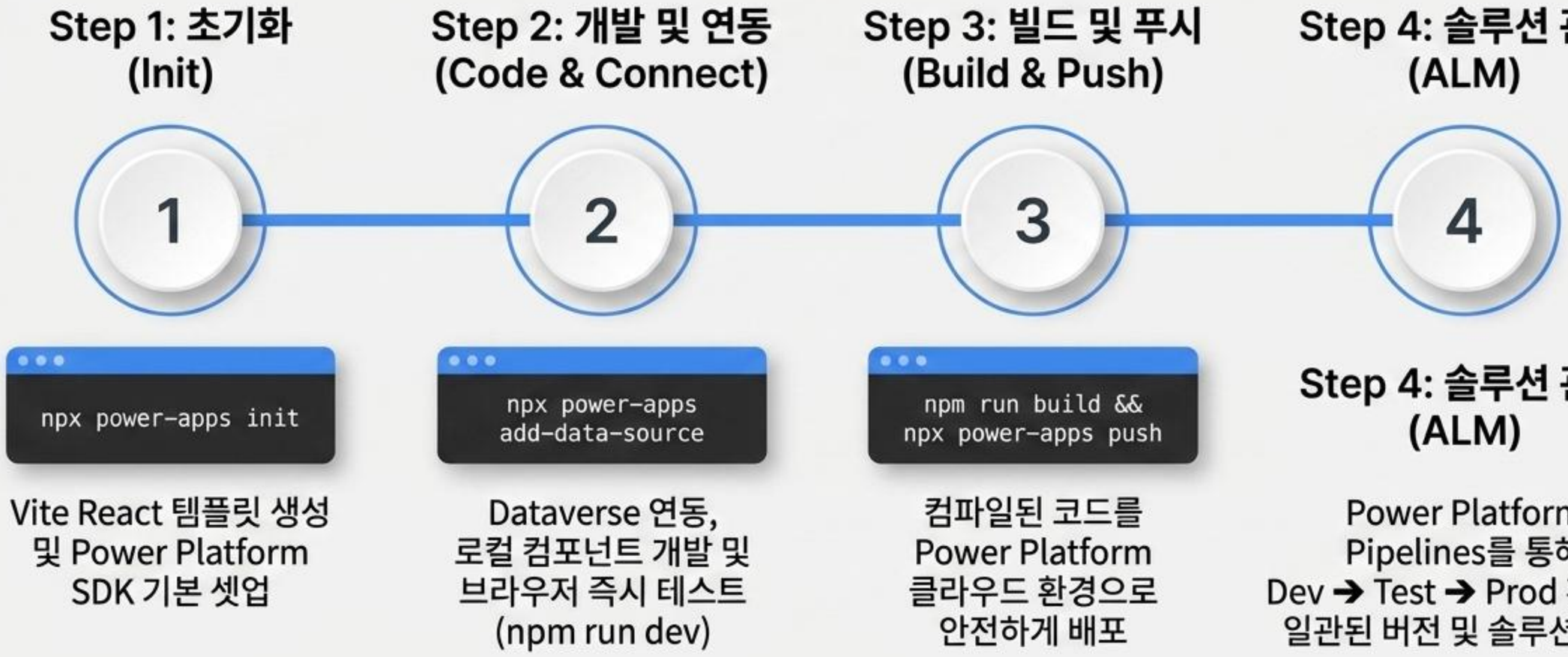
## 엔터프라이즈 데이터 (Dataverse)

관계형 데이터 관리, 메타데이터 접근, DLP(데이터 손실 방지) 정책이 완벽히 적용되는 강력한 보안 백엔드를 제공합니다.

## 1,500+ 커넥터 통합

Azure OpenAI, SharePoint, SQL 등 사내외 시스템 및 Power Automate 워크플로우와 즉각적인 연동을 지원합니다.

# 프로코드 워크플로우 & ALM



# 타협 없는 최적의 개발 경험

**Code Apps는 Low-Code의 민첩성과**
**Pro-Code의 정밀함을 완벽하게 연결합니다.**

## 개발자 경험

- 완벽한 코드 통제력
- 모듈화된 UI 컴포넌트 재사용
- AI (Copilot) 주도 프론트엔드 개발

## 엔터프라이즈 비즈니스

- 인프라 관리 제로(Zero)
- 기존 데이터 자산 (Dataverse) 완벽 통합
- 엔터프라이즈급 보안 및 거버넌스 자동화

# Power Apps 코드 앱 개발, 무엇으로 움직이는가?

기존 캔버스 앱의 제한된 컨트롤과 로직을 넘어, Pro Code 환경은 전 세계 표준 웹 개발 도구를 엔진으로 사용합니다.

## React : 프론트엔드 UI 구축

화면의 모든 요소를 100% 통제. 서드파티 유틸리티 및 기성 컴포넌트 재사용 극대화.

## TypeScript : 복잡한 비즈니스 로직 처리

Power Fx를 대체하는 JavaScript의 발전형. 유연하고 체계적인 로직 작성 및 사전 오류 탐지.

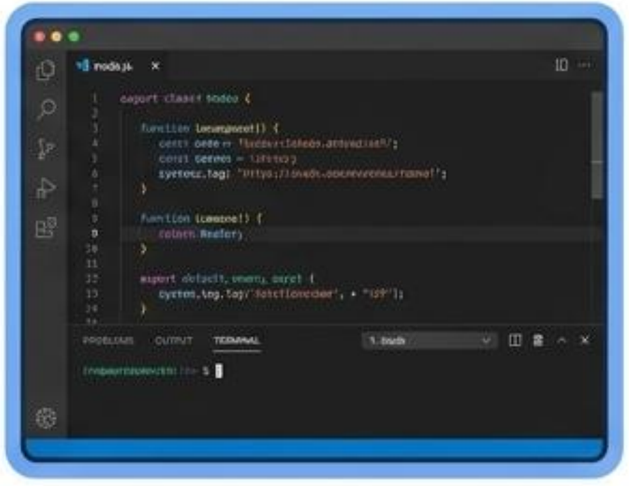

## Vite : 빠르고 효율적인 로컬 구동

코드를 수정하고 저장하면 즉시 화면에 반영되는 HMR 제공. 강력한 프레임워크이자 빌드 도구.

# 필수 설치 도구 3가지 [Prerequisites]

코드를 작성하고 플랫폼에 배포하기 위한 완벽한 로컬 개발 환경 구축 아키텍처

# 로컬 작업 공간의 핵심: VS Code & Node.js



## Visual Studio Code

**역할:** 강력하고 가벼운 무료 코드 편집기 (IDE)

**핵심 기능:** 실시간 화면 테스트(HMR), 터미널 통합, AI 코딩 에이전트 연동

**설치 방법:** 공식 웹사이트 접속 ➡️ OS에 맞는 파일 다운로드 ➡️ 기본 설정으로 설치(Next 반복)

## Node.js

**역할:** 최신 웹 기술 구동 및 패키지 관리기(npm) 제공

**설치 방법:** nodejs.org 접속 ➡️ 반드시 'LTS(안정화)' 버전 다운로드 ➡️ MSI 파일 실행 설치

**[Pro Tip]** 소스 코드 버전 관리를 위해 Git도 함께 설치하는 것을 강력히 권장합니다.

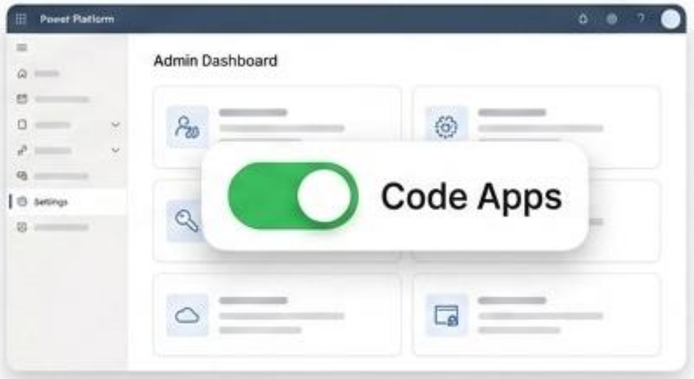
# 가장 핵심적인 연결 고리: Power Platform CLI

로컬 환경과 클라우드 플랫폼을 안전하게 이어주는 '다리' 역할을 수행합니다.


**권장 설치법:** 공식 문서 링크를 통한 Windows MSI 설치 파일 방식이 초보자에게 가장 직관적이고 오류가 적습니다.

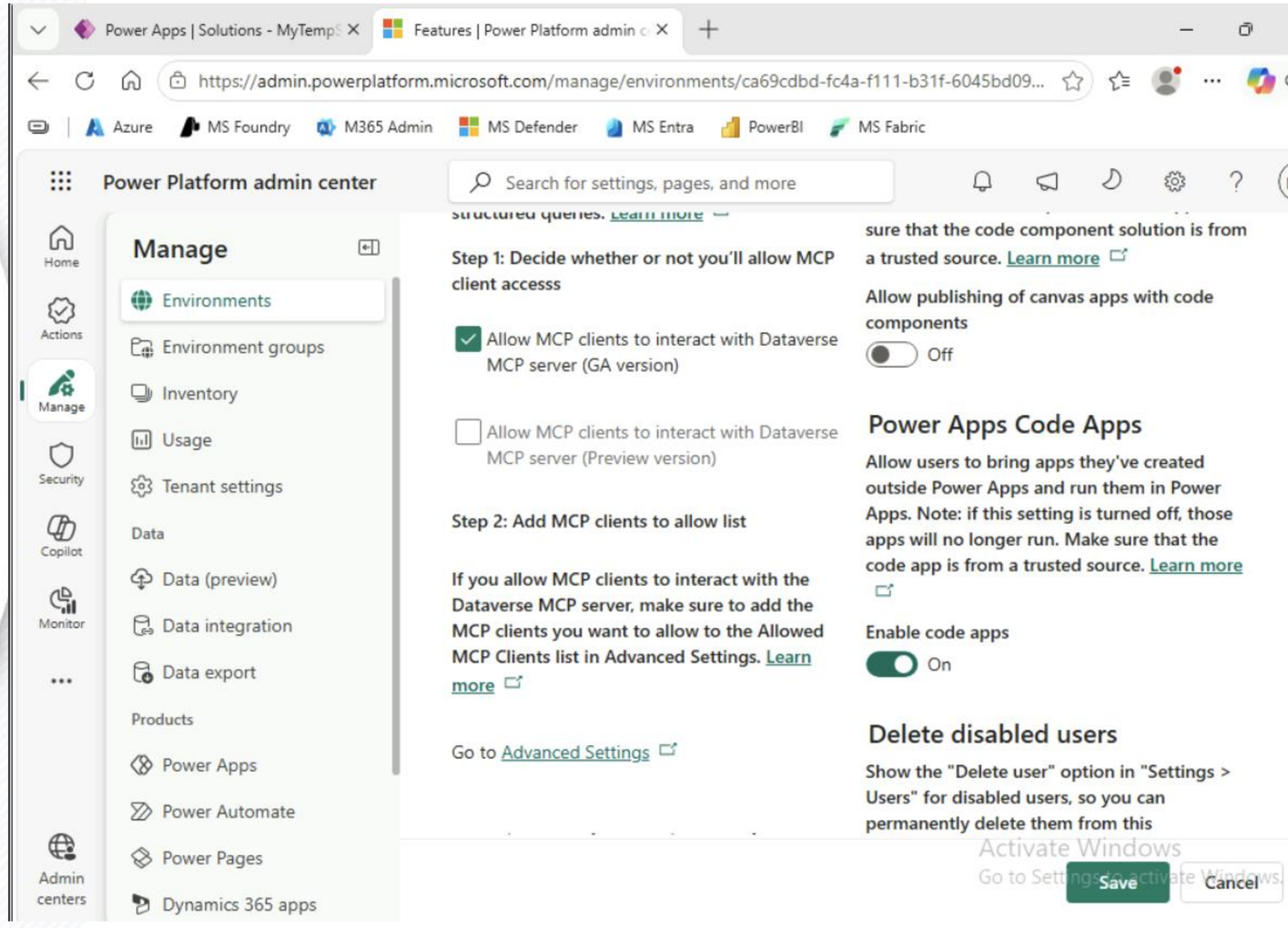
# 잊지 마세요! 'Code Apps' 기능 활성화

내 컴퓨터에 모든 도구 설치가 완벽해도,
플랫폼에서 이 기능이 닫혀있으면 배포할 수 없습니다.




1. **접속**: Power Platform 관리 센터(Admin Center) 접속 → **'환경(Environment)'** 선택
2. **이동**: 우측 상단 **설정**(톱니바퀴) → **'제품 및 기능(Product and features)'**
3. **적용**: 목록에서 'Code Apps' 검색 → **'켜기(On)'**로 변경 후 저장

⚠ 시스템에 완전히 반영되기까지 보통 3~4시간 정도 소요됩니다.
켜자마자 배포 오류가 나더라도 당황하지 마세요!



# Power Apps Pro Code 명령어 라이프사이클

빈 폴더에서 클라우드 배포까지, Visual Studio Code 터미널에서 실행되는 Code Apps 개발의 전체 핵심 명령어 흐름 입니다. 각 명령어는 하나의 파이프라인으로 연결됩니다.

## 준비 단계 (Setup)

1. **npx degit**: 템플릿 뼈대 복제
2. **pac auth create**: Power Platform 계정 인증
3. **npm install**: 필수 패키지 설치
4. **pac code init**: 구성 파일 초기화

## 개발 단계 (Dev & Data)

5. **npm run dev**: 로컬 서버 실행 (HMR)
6. **npx power apps add data source**: 데이터 커넥터 연결 (선택)

## 배포 단계 (Deploy)

7. **npm run build**: 소스 코드 압축
8. **pac code push**: 실제 환경 배포

# 0단계. 명령어 실행 전 필수 도구 준비

본격적인 명령어 실행에 앞서, 로컬 컴퓨터에 아래 3가지 핵심 도구가 반드시 설치되어 있어야 합니다.

## 코드 편집기 (IDE)

터미널을 열고 코드를 작성할 기본 작업 공간.

## 패키지 런타임 (Runtime)

**Node.js (LTS 버전)**

npm 명령어를 사용하고 React/TypeScript 환경을 구동하기 위한 필수 엔진.

## 플랫폼 브릿지 (CLI)

**Power Platform CLI**

pac 명령어를 통해 로컬 앱과 클라우드를 연결 (Windows MSI 설치 권장).

# 1단계. 프로젝트 템플릿 복제 (Bootstrap)

```
> npx degit github:microsoft/
PowerAppsCodeApps/templates/vite
```

가장 먼저 실행할 명령어입니다.
복잡한 초기 설정 없이 즉시 React, TypeScript,
Vite 기반의 웹 개발을 시작할 수 있는
최신 뼈대(템플릿)를 가져옵니다.

# 실시간 테스트의 핵심 원리: HMR

Vite 프레임워크 기반의 HMR(Hot Module Reload)은 코드를 한 줄 변경할 때마다 놀라운 효율을 발휘합니다.

## 전통적 방식 (Traditional)


※ 속도: 느림, 흐름 끊김

## HMR 기반 코드 앱 (Modern)

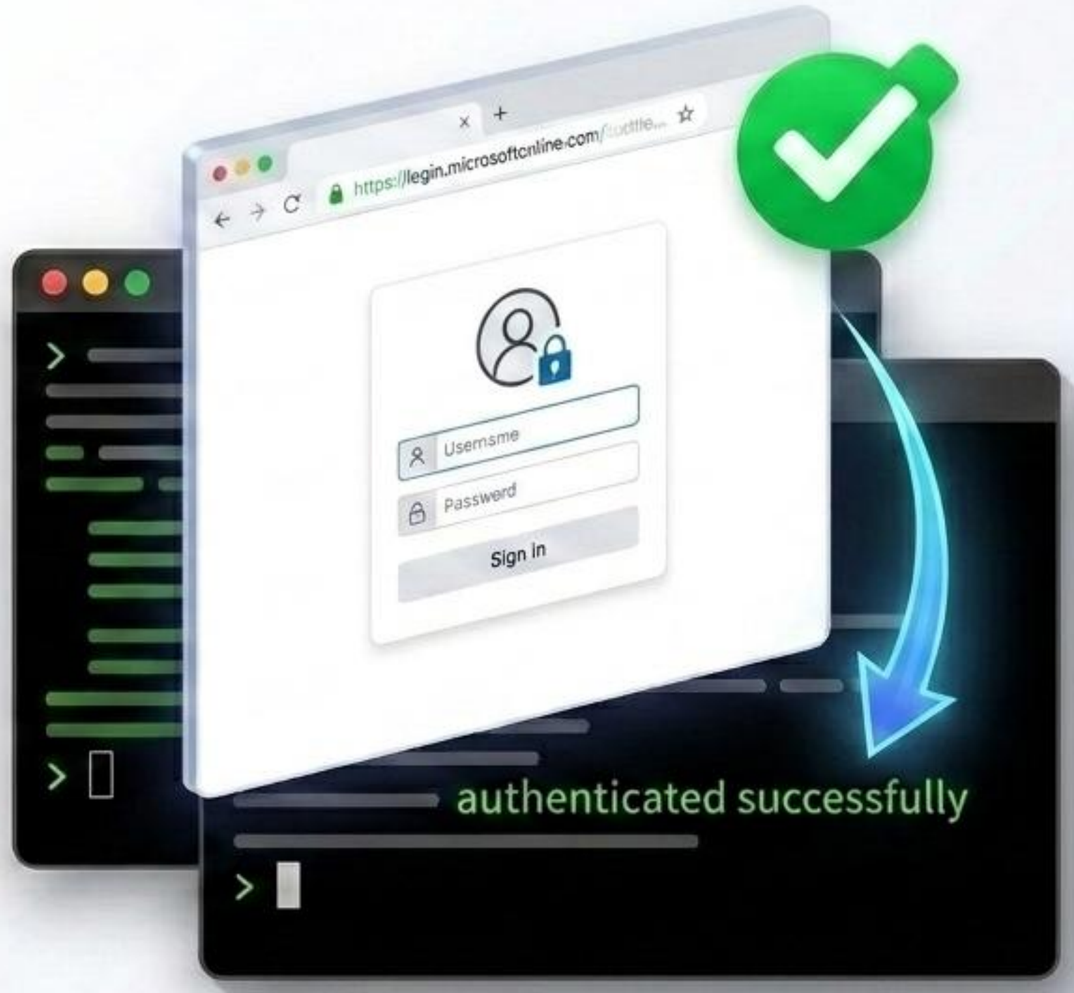

※ 브라우저의 전체 새로고침 없이 변경된 UI 파트만 실시간 교체되며 입력된 데이터 상태가 그대로 유지됨. 개발 속도 극대화.

# 2단계. 플랫폼 인증 및 연결

```
> pac auth create
```

내 컴퓨터에서 작성한 코드를
Power Platform으로 보내기 위해 '내가
누구인지'를 증명하고 통신 다리를 연결합니다.

(참고) 이 단계는 선택 사항입니다. 뒤에서 다룰
npx power-apps init (또는 pac code init)
명령 실행 시 인증 절차가 자동으로 진행되므로,
지금 반드시 수행할 필요는 없습니다.



# 3단계. 필수 패키지 및 종속성 설치

```
> npm install
```

뼈대만 있는 프로젝트에 생명을 불어넣는 과정입니다. 앱 구동에 필요한 외부 모듈과 Power Platform 전용 클라이언트 라이브러리들을 자동으로 다운로드하고 조립합니다.

# 4단계. 코드 앱 공식 초기화

설치된 프로젝트를 내 Power Platform 환경의 '코드 앱'으로 공식 선언하고, 통신 규칙을 정의합니다.

```
> npx power-apps init
```

### 시스템 요청: Environment ID 입력

Environment ID 입력 프롬프트 발생.

### 인증 처리

브라우저를 통한 Microsoft 계정 인증 진행.

### 시스템 요청: Display Name 입력

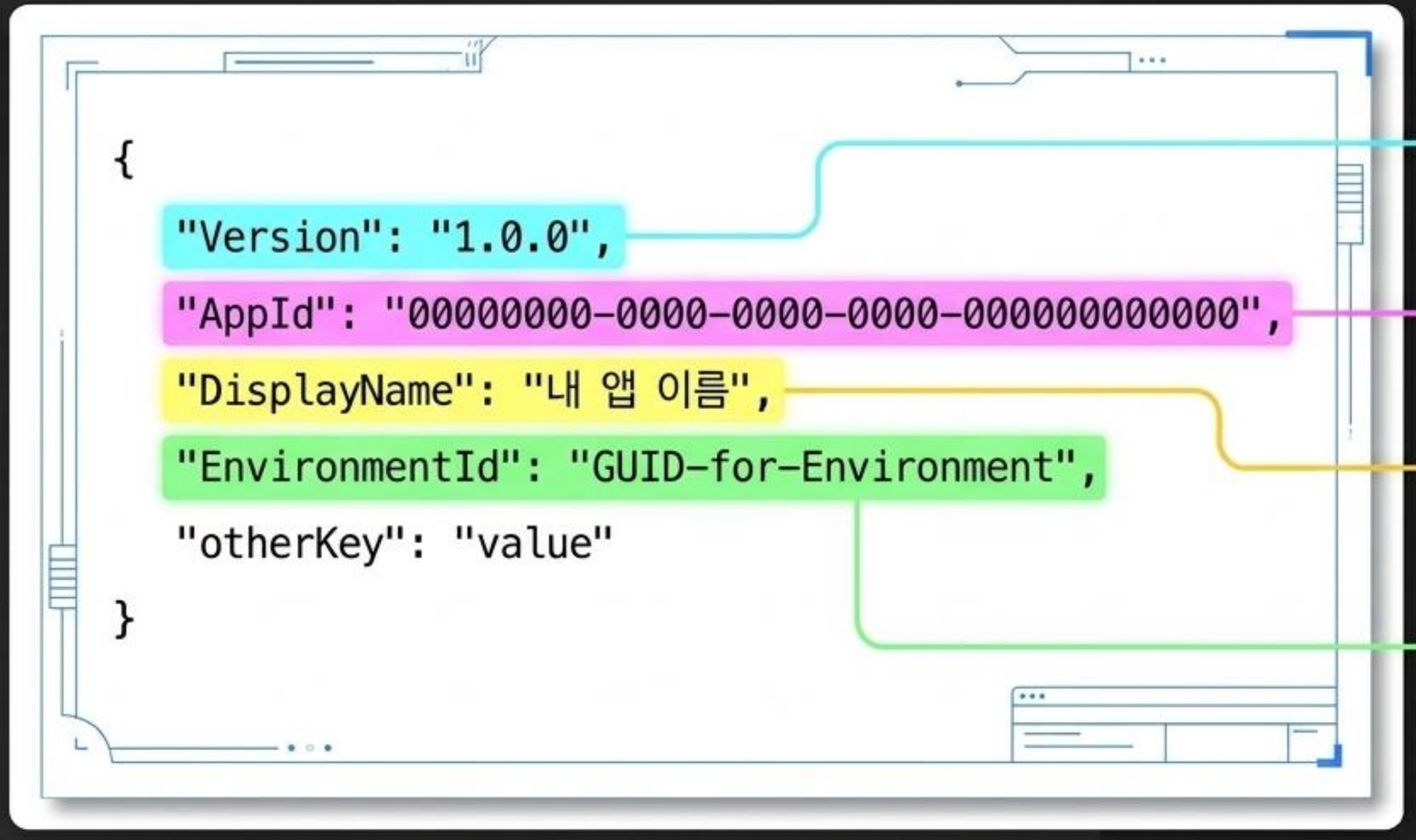
Display Name for My App 입력 프롬프트 발생. (원하는 이름 부여, 영문 권장)

### 최종 생성물: powerapps.config.json 파일 생성

로컬 앱과 클라우드를 이어주는 핵심 설계도 완성.

# 초기화 결과물: 구성 파일 파헤치기

pac code init 실행 후 생성된 powerapps.config.json 파일은 클라우드 관리 센터를 이어주는 설계도입니다. 설정을 변경하려면 이 파일의 값을 직접 수정해야 합니다.




Version : 현재 빌드된 앱의 버전 정보

AppId : 최초 배포 시 고유 GUID가 자동으로 발급됨 (생성 직후엔 빈칸)

DisplayName : 명령어로 지정했던 앱의 노출 이름

EnvironmentId : 앱이 배포될 Power Platform의 목적지 주소

# 5단계. 로컬 개발 서버 구동 (실시간 테스트)

```
> npm run dev
```

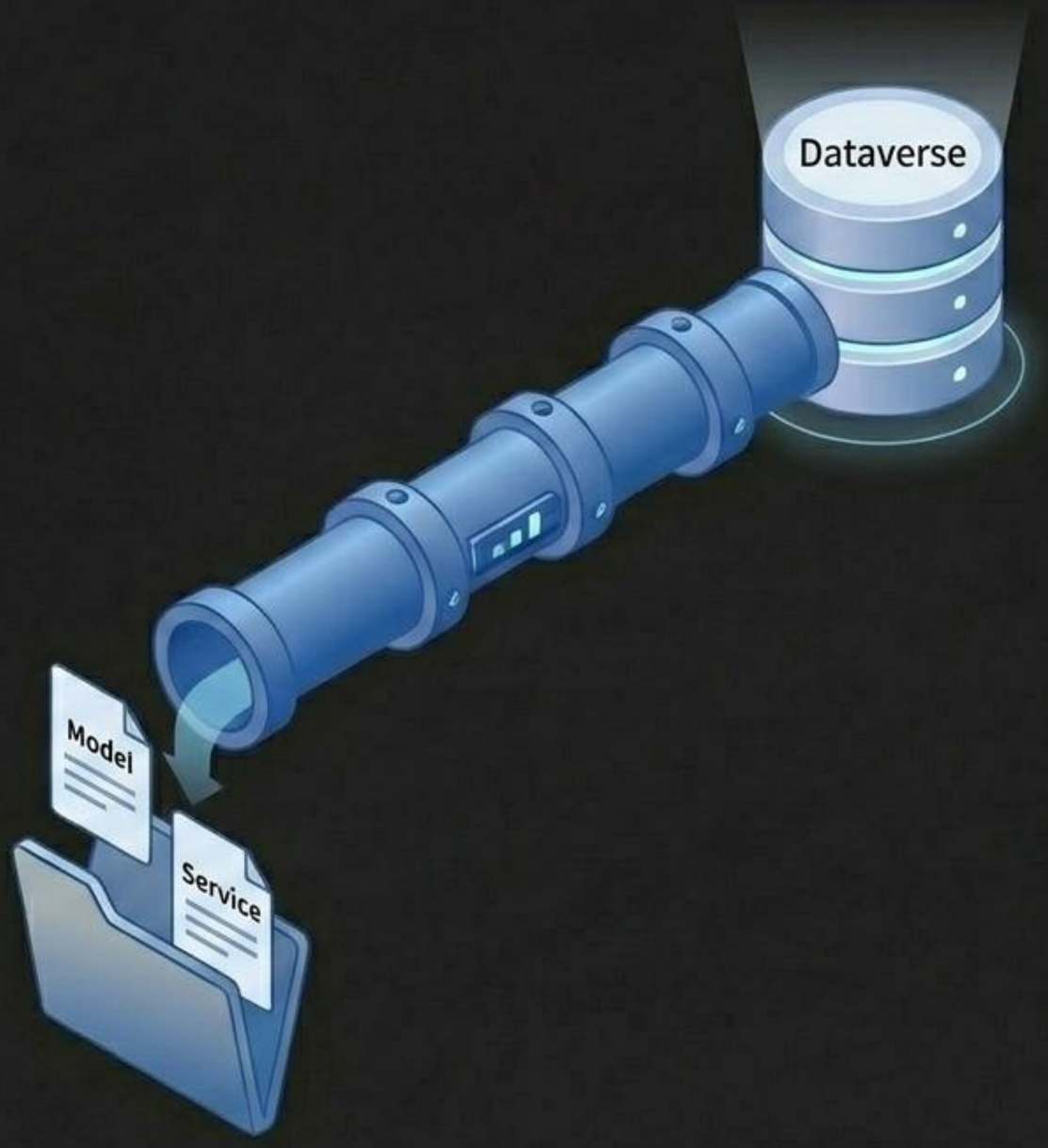
클라우드에 접속하지 않고도 내 컴퓨터 안에서 독자적으로 프론트엔드 UI를 렌더링하고 테스트하는 개발 전용 서버를 띄웁니다.

터미널에서 Ctrl + C를 눌러 서버를 중지할 수 있습니다.

# 6단계. 외부 데이터 소스 연결 (선택)

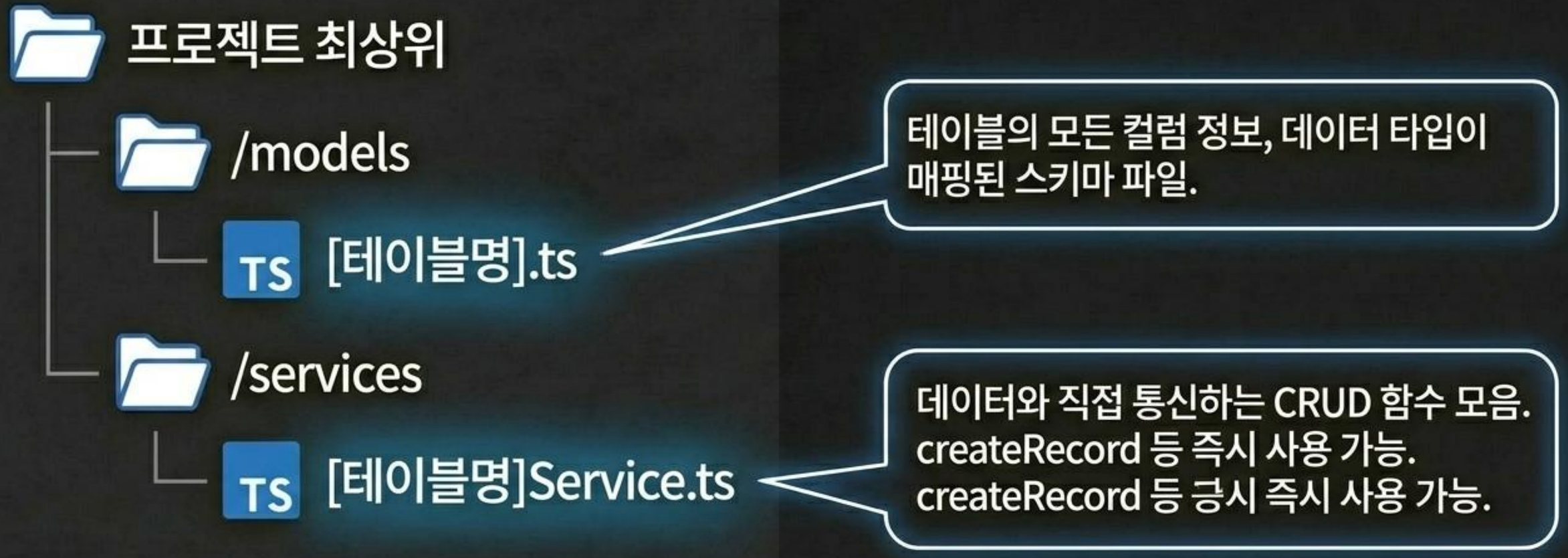
```
> npx power apps add data source
  [조직 URL] [논리적 이름]
```

내 앱에 Dataverse 테이블 등 Power Platform의 실제 데이터를 연결하여 생명력을 불어넣는 과정입니다.



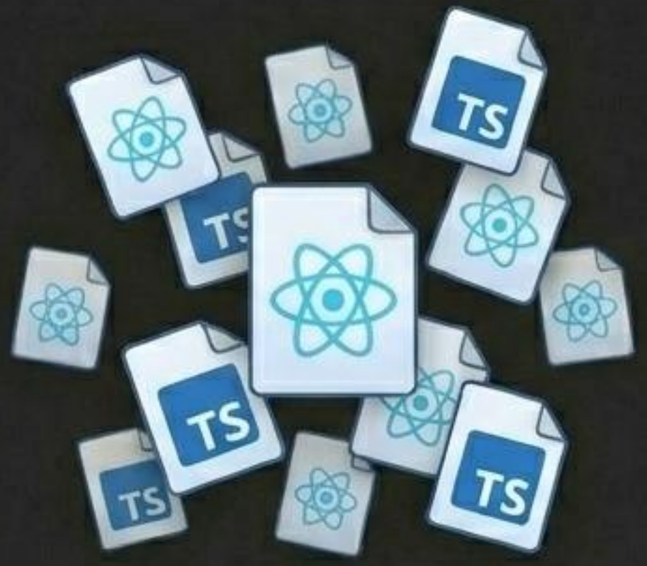
# 데이터 소스 명령어의 결과물

명령어가 성공하면, 즉시 호출 가능한 TypeScript 기반의 구조화된 파일들이 마법처럼 생성됩니다.
자동 완성(IntelliSense)을 완벽하게 지원합니다.



# 7단계. 서버 배포용 파일 생성 (Build)

```
> npm run build
```





원본 코드를 브라우저가 읽기 가장 좋은 순수 JS, HTML, CSS로 변환하고 불필요한 공백을 압축하여 로딩 속도 최적화.

명령어가 완료되면 프로젝트 최상단에 배포 준비가 끝난 결과물들이 담긴 dist (Distribution) 폴더가 새롭게 생성됨.

# 8단계. 코드 앱 최종 배포 (Push)


```
> npx power-apps push
```

웹 호스팅은 이런 고, 마라입아크름정이 따늠 유식한 업습니다. ᵹ 호스팅을 늪격 취압하고 있지문~망면 해 코로베이도 업메트는 3l키프로 찬상마도 보틍된 다스팀 업데이트을 갈정 파압에서 코드를 단죽하여 드립니다. 코스팀을 적용하여, 업데이트를 위해 -Alt T, 확용이 문차되어언에도할 수 있습니다.

# 배포 완료! Power Apps 포털 확인

명령어가 완료된 후 포털(make.powerapps.com)의 '앱(Apps)' 메뉴에 접속하면
새로운 앱을 확인할 수 있습니다.


## ▶ 재생(Play)

기존 앱처럼 브라우저에서 바로 실행되고
사용자들과 공유 가능.

## ⚠ 편집(Edit) 직접 불가

포털 화면 내에서 마우스 드래그 앤 드롭으로
직접 수정 불가. 반드시 로컬 VS Code에서 코드를
수정한 뒤 다시 build 및 push를 실행해야 함.

# 요약: 한눈에 보는 Pro Code 8단계 명령어

지금까지 살펴본 Power Apps 프로 코드 기반 개발의 필수 명령어 흐름입니다.
이 파이프라인을 통해 무한한 자유도를 가진 개발을 시작해 보세요!

✅ npx digit -> 프로젝트 뼈대 생성
✅ pac auth create -> 시스템 인증 연결 (옵션)
✅ npm install -> 필수 부품 설치
✅ npx power-apps init -> 환경 설정 파일 생성
✅ npm run dev -> 로컬 서버 시작 및 HMR 테스트
✅ npm run build -> 배포용 코드 최적화 및 압축
✅ npx power-apps push -> 클라우드 서버 최종 배포 완료

Pro Code & CLI Workflow

# Visual Studio Code에서 SharePoint List 완벽하게 연동하기

Power Apps Pro Code 개발자를 위한
단계별 데이터 소스 통합 가이드

# 클라우드와 로컬 환경을 연결하는 5단계 파이프라인

1

## 1. 연결 생성

(Power Apps 포털)

SharePoint 서비스 인증

2

## 2. ID 확인

(VS Code 터미널)

pac connection list 실행

3


## 3. 소스 추가

(VS Code 터미널)

pac code add-data-source 실행

4


## 4. 파일 자동 생성

(프로젝트 폴더)

Models & Services 구성

5

## 5. 데이터 렌더링

(React 코드)

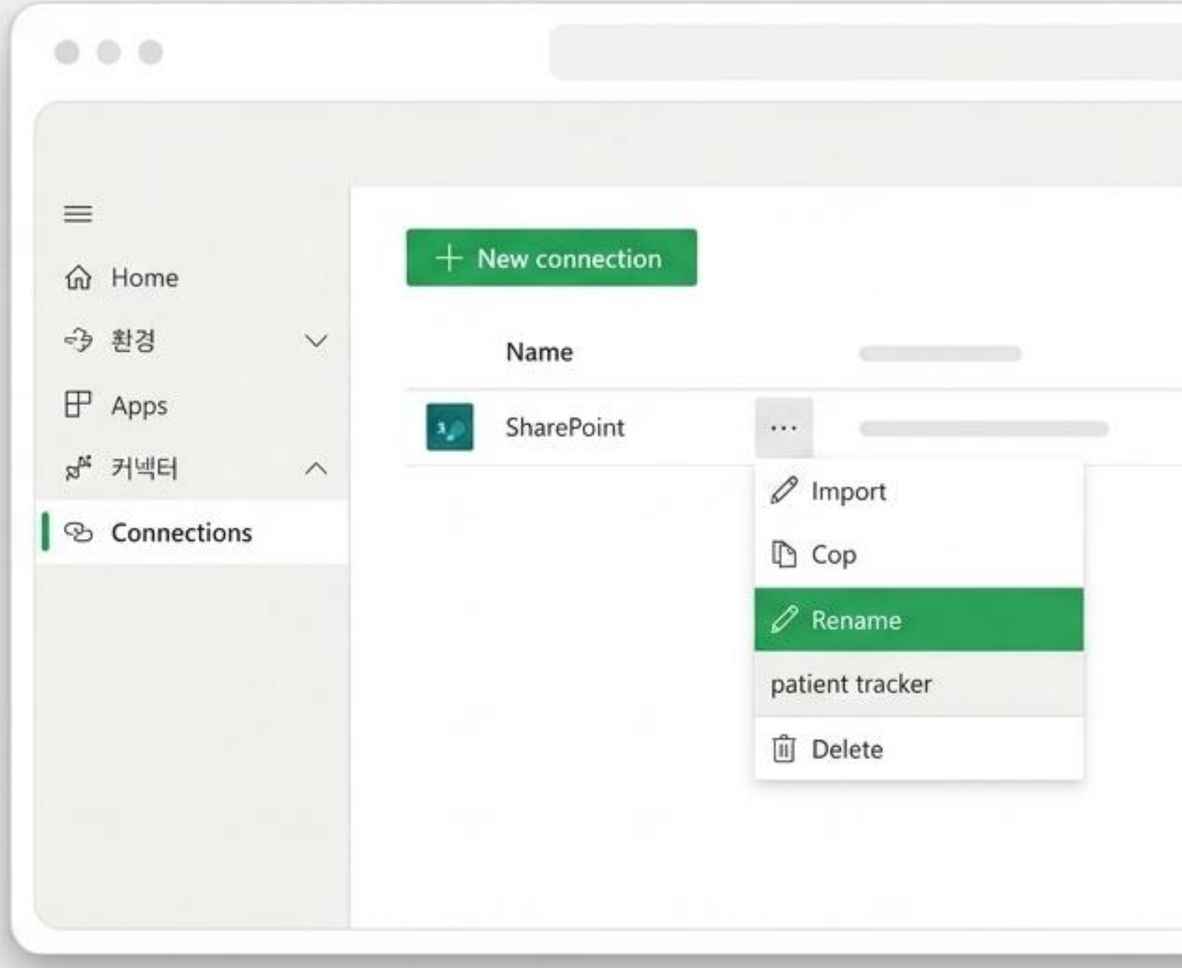
app.tsx에 UI 바인딩

# Step 1. Power Apps 포털에서 안전한 연결(Connection) 생성하기

VS Code에서 데이터를 호출하기 전, 환경 내에 SharePoint와 통신할 수 있는 인증된 연결 고리를 먼저 준비해야 합니다.

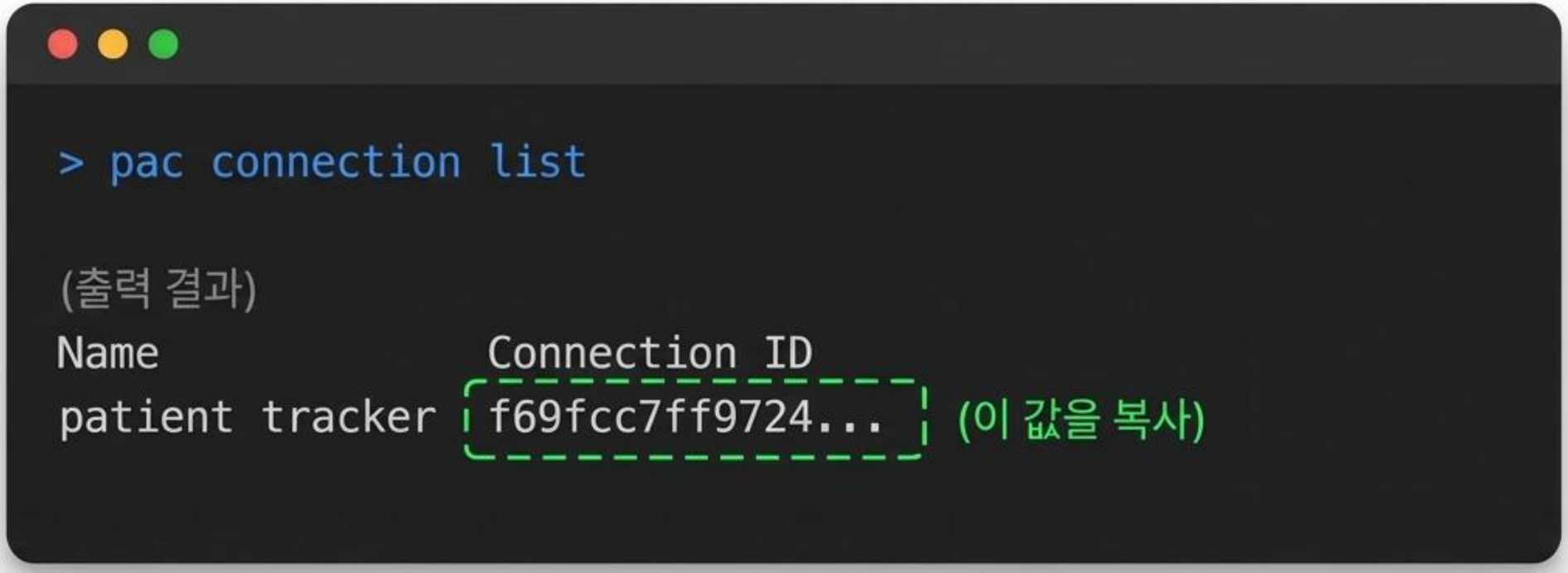
- **메뉴 이동**: Power Apps 메이커 포털 좌측 [Connections] 탭 클릭
- **커넥터 추가**: [+ New connection] 클릭 후 SharePoint 검색 및 선택
- **클라우드 인증**: 'Cloud services' 옵션으로 계정 로그인

**Tip:** 생성된 연결 이름 옆 점 3개를 눌러 식별하기 쉬운 이름(예: patient tracker)으로 변경하세요.



# Step 2. 터미널에서 고유한 연결 ID(Connection ID) 식별하기

VS Code 터미널에서 명령어를 실행하여 1단계에서 명명한 연결의 고유 주소(ID)를 찾아 복사합니다.




포털의 연결 상세 페이지 URL 끝부분에서도 동일한 연결 ID를 확인할 수 있습니다.

# Step 3. 4가지 매개변수를 활용한 데이터 소스 주입 명령어 실행

```
pac code add-data-source -a [API] -c [ID] -t [Table] -d [URL]
```

### -a (API 이름)

shared_sharepointonline
(SharePoint 기본값으로 고정)

### -c (Connection ID)

Step 2에서 복사해 둔 고유 연결 ID 문자열

### -t (Table Name)

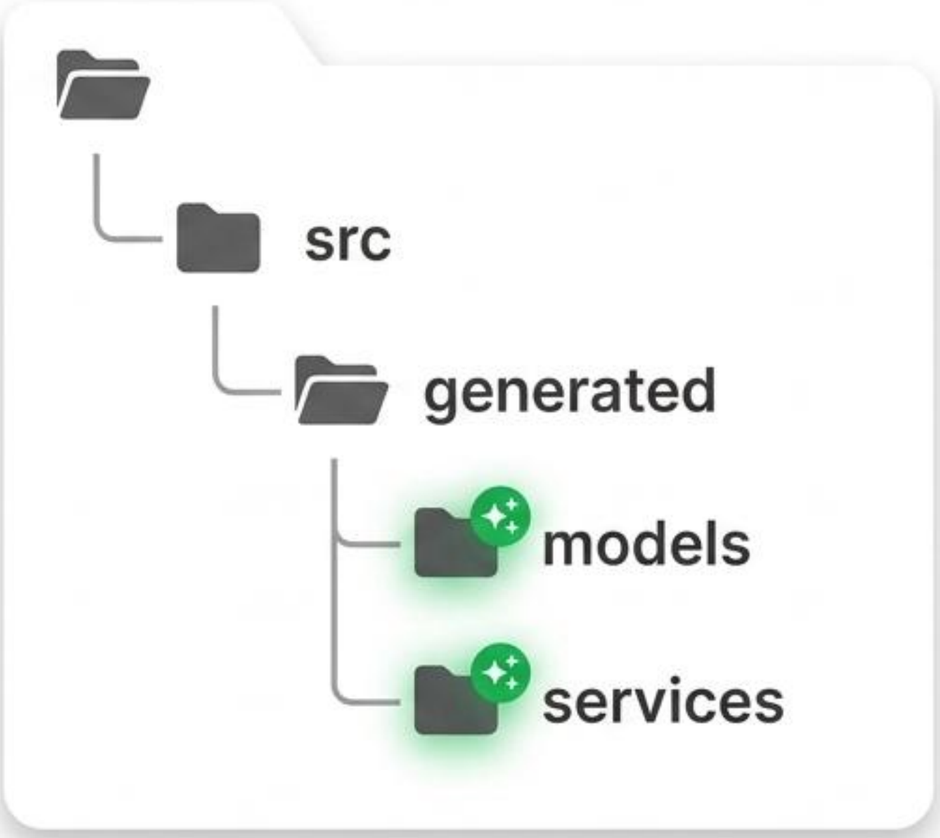
SharePoint List의 이름
(띄어쓰기가 있다면 URL 상의 %20을 그대로 포함)

### -d (Data URL)

대상 데이터가 위치한 SharePoint 사이트의 기본 웹 주소

# Step 4. 백그라운드에서 자동 생성되는 보일러플레이트 구조 이해하기

명령어가 성공하면 통신 로직을 대신 처리해 줄 **두 개의 핵심 폴더**가 즉시 생성됩니다.
**직접 통신 코드를 짤 필요가 없습니다.**




### models 폴더 (TypeScript Interfaces)

SharePoint List의 스키마 구조를 자동으로 읽어와 모든 열(Column)을 타입(Type)으로 정의합니다. 강력한 타입 안정성을 제공합니다.

### services 폴더 (CRUD Methods)

데이터를 조회하고 수정하기 위한 네이티브 **API** 메서드(예: **getAll()**)가 내장된 클래스 파일입니다. 복잡한 **HTTP** 통신을 완벽히 추상화합니다.

# 복잡한 React 코드, 일상 언어(자연어)로 지시하다

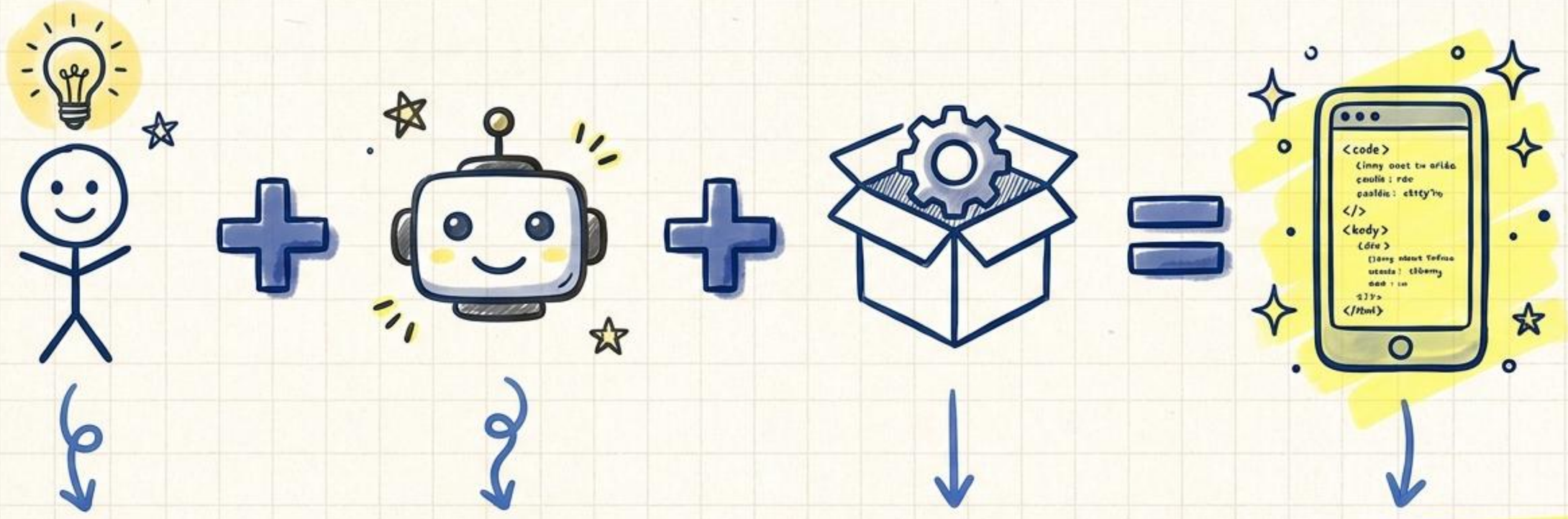
타이핑 없이 프롬프트만으로 원하는 기능을 완벽하게 구현하는 AI 에이전트의 마법.

Github Copilot
& Claude
활용 비법 노트

# AI와 VS Code로 시작하는 마법의 Code App 개발

천재 개발자의 비밀 코딩 노트

# 새로운 개발 공식



아이디어와 기획

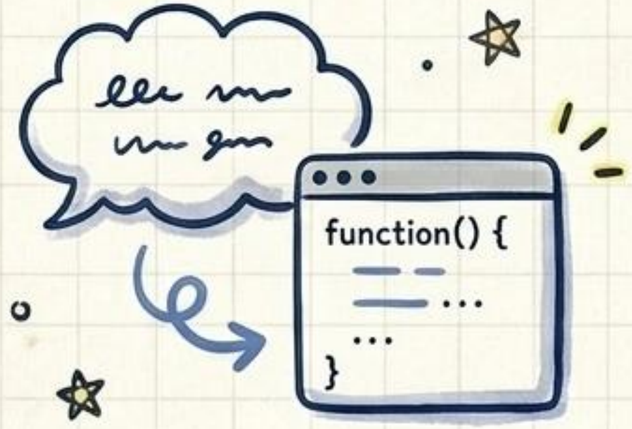
자연어로 지시
(프롬프트)

로컬 코드 편집기

100% 맞춤형 웹 앱
(React/TypeScript)

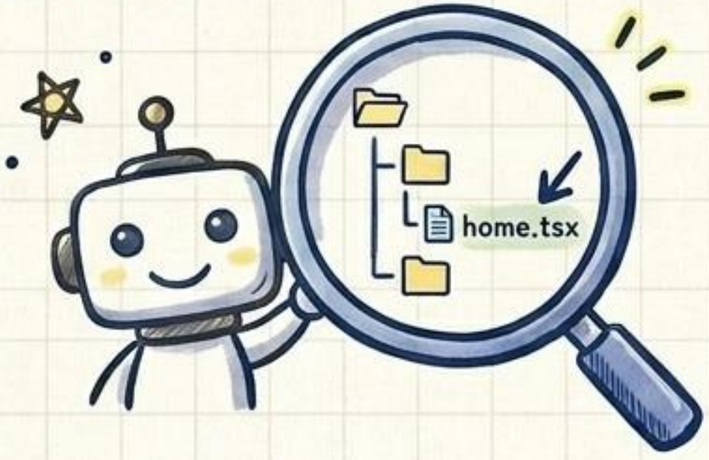
# AI 에이전트의 3가지 초능력

## ① 자연어 코드 생성



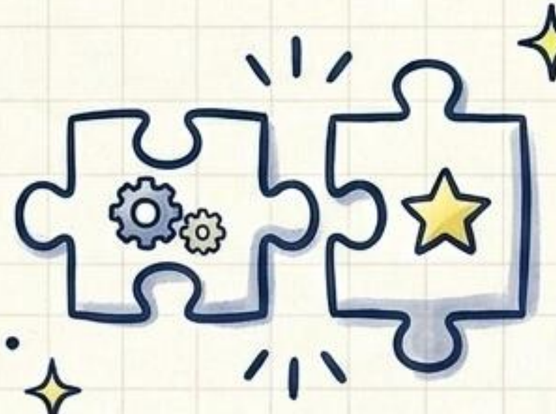

칸반 보드 만들어줘!
일상 언어로 지시하면
AI가 복잡한 코드로 자동 변환.

## ② 프로젝트 구조 완벽 이해




어디부터 편집할까?
VS Code 내의 복잡한 파일 구조를
핵심 파일(home.tsx)을 스스로 탐색.

## ③ 기존 맥락 파악 지능



기존에 구현된 기능을
인지하여 중복을 피하고,
스마트하게 새로운 기능에 집중.

# AI Code App의 명암 (Pros vs Cons)

## 장점

- 압도적 시간 단축: 복잡한 UI(칸반 보드 등)를 단 10분 만에 MVP로 완성.
- 로우코드 한계 극복: 부드러운 애니메이션, 드래그 앤 드롭 완벽 구현.
- 엔터프라이즈 보안: Microsoft Entra ID 및 DLP 정책 등 강력한 사내 보안 유지.

## 단점

- 시각적 편집기 부재: 브라우저 포털에서 마우스로 직접 화면 수정 불가.
- 프리미엄 라이선스 요구: 배포 후 앱을 실행하는 최종 사용자는 모두 프리미엄 라이선스 필수.

※ 버그 수정이나 기능 업데이트 시 반드시 VS Code로 돌아와서 코드를 고쳐야 함!

# 실전 운영 환경 배포 필수 가이드

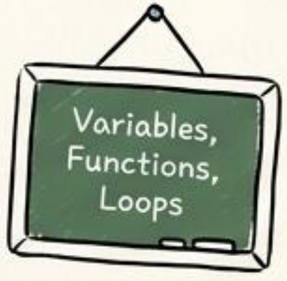
## To-Do List

### ☐ 전문 개발자 코드 리뷰 필수

AI가 짠 MVP 코드는 훌륭한 초안일 뿐.
완벽히 이해하지 못하는 코드를 실제 운영 환경에
배포하는 것은 절대 금물! 기업 표준 검증은 필수입니다.


### ☐ 기초 프로그래밍 지식 함양 ★

AI 코드를 장기적으로 유지보수하려면 시민 개발자도
변수, 함수, 반복문 등 최소한의 프로그래밍 원리를 이해해야 합니다.



Claude와 Copilot은 완벽한 초안 작성자(Draftsman)

결과물의 안전성과 완성은 인간의 통제(Human Control)

AI와 VS Code가 시민 개발자와 전문 개발자의 경계를 허물다.

# 후원사

## 다이아몬드

Microsoft

## 플래티넘

Korea Software Service

(주)한국소프트웨어서비스

## 골드

go deploy

STK

TRAINOCATE

## 실버

OPEN SG Solution Group

## 브론즈

한빛미디어