# 김지욱

깃허브 블로그

**백엔드를 중심으로, 필요한 화면까지 직접 만들어 제품을 완성하는 5년 차 개발자입니다.**

- **PostgreSQL + NestJS** API 서버를 직접 설계·구축하고, AWS에서 배포부터 모니터링·장애 대응까지 백엔드 전반을 맡아 B2C/B2G 교육 플랫폼을 운영했습니다.
- 운영 지표에서 병목을 찾아 고칩니다. RDS CPU 장애를 쿼리·인덱스 개선으로 해결했고(쿼리 실행 **2.71초 → 5.78ms**), 30명 동시 컴파일 평균 응답 시간을 **120초 → 11.6초**로 줄였습니다.
- 회고 웹앱 **CHECKIN** 등 필요한 제품을 직접 기획·개발·배포하고, Claude Code로 **AWS 관리 도구**를 만들어 실무에 적용해 왔습니다.

## 경력 4년 8개월

### 주식회사 코디플레이

2025.07 – 2026.03 (9개월) · 정규직

#### 코디플레이 스튜디오 신규 서비스 개발 (베타)

2026.02 (1개월) · 풀스택

Claude Code, React, Supabase Edge Functions

| | |
|---|---|
| 문제 인식 | • 기존 블록 코딩 에디터의 **진입점 부재**(학습 콘텐츠 탐색 및 프로젝트 이어서 열기 불가) 문제 해결<br>• 실제 수업 현장에서 학생들의 **회원가입·로그인 허들을 없애고** 즉시 체험할 수 있는 신규 서비스 구축 필요 |
| 해결 과정 | • **Supabase Edge Functions**로 학습 콘텐츠 조회, 프로젝트 조회·포크 API를 개발<br>• **아키텍처·코딩 컨벤션 문서를 정의**하고, 이를 기준으로 Claude Code와 함께 React 화면을 기능 단위 모듈로 구현<br>• 로그인 도입 전 단계에서는 **기기 ID로 사용자를 식별**해, 인증 없이도 바로 써볼 수 있게 설계 |
| 성과 | • **한 달 만에** API부터 화면, 배포까지 갖춘 베타 서비스 오픈<br>• 학습 콘텐츠 탐색 → 새 프로젝트 생성 → 에디터 이동을 **하나의 흐름으로 연결**해, 로그인 없이 바로 시작할 수 있는 진입점 마련 |

#### 서비스 인프라 분리 및 비용 최적화

2026.01 – 2026.02 · 백엔드/인프라

AWS (EC2, RDS, Lambda, VPC, CloudWatch), Claude Code

| | |
|---|---|
| 문제 인식 | • 법인 분리 전부터 코디플레이 AWS 계정 안에 **마로마브 서비스 리소스와 공용 리소스가 섞여** 있어, 서비스별 비용 파악과 독립 운영이 어려움<br>• 서버 스펙이 실제 사용량보다 크게 잡혀 있고, **쓰지 않는 리소스가 비용을 계속 발생**시킴 |
| 해결 과정 | • 공용 리소스의 **서비스별 의존 관계**를 파악해 계정별로 분리하고, **병행 운영으로 검증**한 뒤 전환<br>• 미사용·고비용 리소스를 정리하고, EC2를 **수업 스케줄의 피크를 감당하는 최소 사양**으로 조정<br>• 호출량이 적은 내부·베타 서비스는 **서버리스**로 전환해 무료 사용량 안에서 운영<br>• Claude Code 기반 AWS 관리 킷(**infra-as-claude**)을 직접 개발해 반복 운영 작업을 자동화 |
| 성과 | • 코디플레이 계정 월 AWS 비용 **100만원 → 50만원** (분리 전 2개월 vs 분리 후 2개월) |

### Arduino 컴파일 서버 Node.js → Go 마이그레이션

2025.11 – 2025.12 · 백엔드

Go, Gin, Arduino CLI, Docker, Claude Code

| | |
|---|---|
| 문제 인식 | • 교육용 블록 코딩 플랫폼의 컴파일 서버가 요청마다 arduino-cli 자식 프로세스를 **동시 실행 상한 없이** 실행<br>• 한 반(30명)이 동시에 컴파일하면 프로세스들이 **CPU를 두고 경합**해 평균 응답 시간이 120초까지 늘어남<br>• 수업 시간대마다 **서버를 증설해 버티는** 구조 |
| 해결 과정 | • CPU 바운드 작업이라 Node.js 구조를 손보는 대신, **경량 동시성과 단일 바이너리 배포**가 강점인 Go로 컴파일 서버를 새로 만들기로 결정<br>• **세마포어로 동시 컴파일 수에 상한**을 두어 초과 요청은 대기열에서 순서대로 처리<br>• 기존 Node.js API와 **같은 응답 포맷**을 유지하고, **병렬 운영 기간**을 두어 단계적으로 트래픽 전환<br>• **동일 사양(t3.small) 서버**에서 기존 서버와 동시 요청 결과를 비교해 검증 |
| 성과 | • 30명 동시 컴파일 요청 기준 평균 응답 시간 **120초 → 11.6초** (동일 사양 서버 비교)<br>• 수업 시간대 증설 없이 서버 **2대 → 1대**로 운영 |

### 글로벌 개발팀 인프라 운영 및 커뮤니케이션

2025.07 – 2026.03 · 백엔드/인프라

- 한국·카자흐스탄·러시아 개발자들이 참여하는 글로벌 팀의 AWS 인프라를 단독으로 담당
- Prometheus + Grafana로 서버 지표 수집·대시보드 구성
- 원격으로 떨어진 팀원 간 교류를 위해 월간 개발자 미팅을 기획·운영

## 주식회사 마로마브

2021.08 – 2025.07 (4년) · 정규직

### 공유 프로젝트 조회 API 슬로우 쿼리 개선 (RDS CPU 100% 장애 대응)

2022.12 · 백엔드

PostgreSQL, TypeORM, NestJS, AWS RDS, CloudWatch

| | |
|---|---|
| 문제 인식 | • 서울·인천 출강 수업이 몰리는 오전 시간대에 **RDS CPU가 100%에 도달**하고 서버가 응답하지 못하는 상황이 반복됨<br>• **EXPLAIN ANALYZE**로 공유 프로젝트 목록 API를 병목으로 특정: 좋아요·다운로드 수 집계가 **인덱스 없이 테이블을 전체 스캔**하고, 페이지네이션이 조인 결과 전체를 정렬한 뒤에 적용됨 |
| 해결 과정 | • 집계 대상 FK 컬럼( `likes.user_project_id` , `download_logs.user_project_id` )과 `is_shared` 에 **인덱스 추가**<br>• 페이지에 해당하는 프로젝트 id만 **서브쿼리로 먼저 뽑은 뒤 조인**하도록 쿼리 구조 변경 |
| 성과 | • 쿼리 실행 시간 **2.71초 → 5.78ms** (EXPLAIN ANALYZE 기준)<br>• API 응답 시간 **1.48초 → 0.14초** (단건 요청 기준)<br>• 매일 오전 반복되던 **RDS CPU 100% 피크가 사라지고 10% 내외로 안정** (CloudWatch, 배포 전후 1주 비교) |

### 교육청 소속 판별을 위한 DB 구조 개선

2023.06 · 백엔드

PostgreSQL, TypeORM, NestJS

| | |
|---|---|
| 문제 인식 | • 서울·인천 교육청 MS 로그인 도입 후, 소속 교육청을 판별하려면 users → departments → managers → ministry_tenants를 **이메일 도메인 LIKE 매칭으로 조인**해야 함<br>• 유저 조회 API 응답이 **4초를 넘고**, 교육청별 이용 통계 쿼리도 복잡해짐 |
| 해결 과정 | • 교육청을 `ministries` 테이블로 분리해 문자열 중복을 없애고, **교육청이 추가되어도 스키마 변경 없이** 대응할 수 있게 설계<br>• `users` · `managers`에 `ministry_id` FK를 두어 **소속을 직접 조회**하도록 변경하고, 기존 데이터는 SQL로 백필 |
| 성과 | • 유저 조회 API 응답 시간 **4.06초 → 124ms** (단건 요청 기준)<br>• 교육청별 유저 집계를 **단일 조인**으로 조회 가능 |

### Firebase → PostgreSQL 마이그레이션 및 API 서버 재구축

2021.09 – 2021.12 · 백엔드

NestJS, TypeORM, PostgreSQL, Firebase

| | |
|---|---|
| 문제 인식 | • Firebase(NoSQL) 기반으로 운영되던 서비스에서 **복잡한 관계형 데이터 처리의 한계**가 명확해짐<br>• 별도 API 서버 없이 **클라이언트가 Firebase에 직접 접근**하는 구조로 확장성 문제 발생 |
| 해결 과정 | • **Firestore 컬렉션 10개를 관계형 스키마로 재설계**: 문서 안 배열(좋아요 유저 목록)은 조인 테이블로, 학급의 교사 목록 배열은 FK로 분리하고, 원본 문서 ID는 `legacy_id` 컬럼에 보존해 관계 매핑과 추적에 사용<br>• **NestJS** 기반 RESTful API 서버 구축<br>• 주말 사용량이 거의 없는 교육 서비스 특성상, 무중단 이관의 복잡도보다 **주말에 공지 후 서비스를 중단하고 일괄 이관**하는 편이 위험이 적다고 판단. 이관 스크립트 전 과정에 로깅을 적용해 실패 건을 추적·재처리 |
| 성과 | • 클라이언트 직접 접근 구조를 **API 서버 경유 구조**로 전환해 비즈니스 로직과 권한을 서버에서 통제<br>• 이후 교육청 소속 판별, 교육청별 이용 통계 등 **B2G 고객 요구에 맞춘 기능 확장**의 토대 마련 |

### 웹/앱 서비스 백엔드 개발 및 인프라 운영

2022.01 – 2025.07 · 백엔드

- 교육청·학교와 개인 사용자를 대상으로 하는 B2C/B2G 교육 플랫폼의 백엔드 API 개발과 AWS 인프라를 구축·운영
- GitHub Actions + AWS CodeDeploy를 활용한 무중단 자동 배포 파이프라인 구성
- CloudWatch 지표·알람을 Slack과 연결해 서비스 상태를 모니터링하고 장애에 대응
- 사내 발표 문화(24절기록)를 기획·운영해 개발·디자인 등 다양한 주제의 정기 발표를 진행

## 스킬

TypeScript | Node.js | NestJS | TypeORM | PostgreSQL | AWS | GitHub Actions | Docker | Claude Code

## 학력

**백운고등학교** · 인문계 졸업 2008.03 – 2011.03