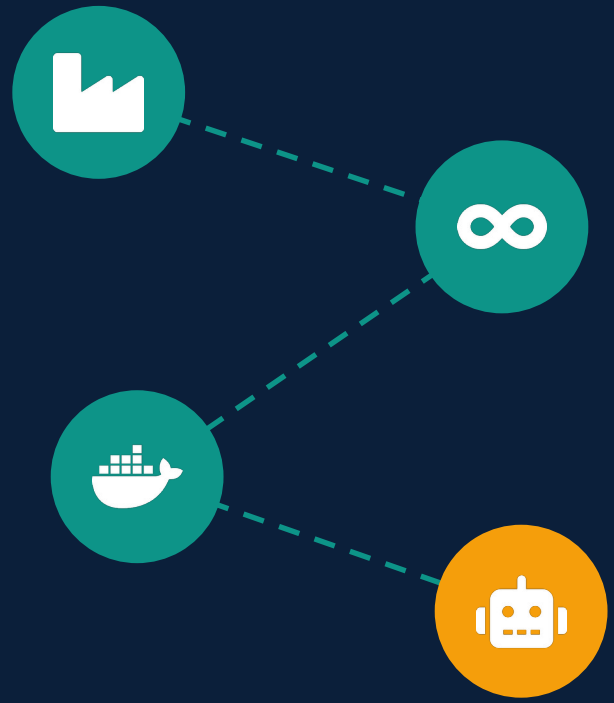
디지털 트랜스포메이션 · DevOps · Docker — 2026 개정판

AI 시대의 디지털 전환, DevOps 그리고 컨테이너

바이브 코딩과 코딩 에이전트까지 현행화



채문창 · 원본 → 2026 개정

발표에 앞서 — 지식이 아닌 큰 줄거리

원본의 첫 질문 세 개를 2026 년에 다시 던집니다



위치

나는 어디에 ?

코드를 직접 쓰는
개발자에서
에이전트에게 일을 맡기고
검증하는 개발자로 바뀌는
중



현황

현재 상황은 ?

개발자 90% 가 AI 를 쓰고
(DORA 2025)
코드 작성 비용은 거의 0 에
가까워짐



방향

어디로 가야 하나 ?

무엇을 만들지 정하고 (명
세)
맞게 만들었는지 확인하는
(검증) 역량으로

원본 이후 , 무엇이 바뀌었나

원본 (2017~2020)

디지털 전환 · 사람이 코딩

- 디지털 트랜스포메이션 (DX)
- DevOps 로 빌드 · 배포 자동화
- Docker 로 개발 환경 표준화

2026 지금

AI 전환 · 에이전트가 코딩

- AI 트랜스포메이션 (AX)
- DevOps + 코딩 에이전트
- 컨테이너 = 에이전트의 안전한 작업장

원본의 결론 ' 가장 비싼 자원 (YOU) 을 최적화하라 ' 는 오히려 더 맞는 말이 되었습니다

오늘의 순서

원본 288 쪽의 흐름 그대로 , 각 장 끝에 2026 년의 답을 붙였습니다



01

발표자 소개

35 년 개발 여정과 기록



02

변화의 연속성

UI · 언어 · 개발 환경



03

DX 에서 AX 로

산업혁명 · 시가총액 · 사례



04

자원과 비용

가장 비싼 자원은 누구인가



05

DevOps 진화

단계별 도구와 DORA 지표



06

컨테이너

클라우드 · 12-Factor · devcontainer



07

무인화 걸림돌

Code · Plan, 그 후



08

마치며

Change · Challenge · Chance

01

발표자 소개

60 살까지 개발하겠다던 약속 , 그 후



35 년의 개발 여정

여섯 번의 직장과 스타트업 , 그리고 지금

1986

BASIC 첫 만남

고교 축제 , '60 살까지 개발 ' 다짐

1990

첫 스타트업

KT 옐로페이지 검색 · NEIS

1996

두 번째

번역관리 · 조달 Web EDI

2001

세 번째

SAP R3 생체인증 보안

2005

네 번째

CEP 상관분석 · Anti-DDoS

2012

첫 직장

제품 + 기술지원 + 사업

2017

두 번째 직장

온프레미스 + SaaS

2026

이즈잇소프트

1 인 기업 · 산업 AI · 스마트팩토리

블로그를 시작한 이유 , 그리고 기록의 새 독자

블로그 기록

나와 남을 위한 기록

- 기억력 감퇴에 대비
- 내가 하는 일의 기록
- 남의 시행착오 시간을 줄이기

2026 기록

에이전트도 읽는 기록

- README · AGENTS.md · ADR 이 에이전트의 기억
- 기록을 잘하는 사람이 에이전트도 잘 쓴다
- 플랫폼은 사라진다 → 내 git 저장소에

기록하는 습관은 AI 시대의 가장 강력한 개발 역량입니다

02

변화의 연속성

기술은 바뀌어도 흐름은 이어진다



UI 의 연속성

화면을 그리는 방식에서 , 말로 시키는 방식까지

1987

CLI · Text UI

명령어와 텍스트 화면

1993

RAD

Delphi · VB · PowerBuilder

1996

Web App

PHP · ASP · JSP

2001

X-Internet

Thick + Thin Client

2005

RIA

Flex · Silverlight

2010

SPA

HTML5 · CSS3 · REST API

2017

PWA

오프라인 · 설치 · 푸시

2024~

대화 · 에이전트 UI

채팅 · 음성 , 에이전트가 앱을 사용

개발 언어의 연속성

마지막 칸의 언어는 ' 자연어 ' 입니다

1987
C
애플리케이션 · 커널

1992
C++
애플리케이션 · 라이브러리

1994
Perl
관리 도구

1997
Java
WAS · JSP

2008
Python
지금까지 주력 (^^)

2012
JavaScript
DOM 위젯 · 프레젠테이션

2015
Go
Docker · InfluxDB · Grafana

2023~
자연어 + 코드
명세는 말로 , 검증은 코드로

개발 환경의 연속성

35 년 만에 다시 터미널로 — vi 에서 시작해 터미널 에이전트로

1991

vi

디버깅 · 텍스트 처리

1996

Visual Studio

RAD, 화면 = 코드

2005

Eclipse

Java · CDT

2010

PyCharm

JetBrains IDE

2021~

IDE + AI 보조

자동완성 · 채팅 · 편집

2025~

터미널 에이전트

claude · codex · agy + tmux

그럼에도 불구하고 변치 않는 것

원본 105~108 쪽의 3 계층은 그대로입니다 . 바뀐 것은 ' 누가 작성하느냐 ' 뿐



에이전트

Presentation

사용자 UI · UX
화면 코드는 에이전트가
가장 잘 만드는 영역



사람

Domain Logic

UI 없는 업무 규칙
도메인을 이해한 사람만이
맞는지 판단할 수 있음



함께

Data Access

저장소 접근
코드는 에이전트가 ,
스키마와 마이그레이션
승인은 사람이

03

디지털 전환에서 AI 전환으로

DX(Digital Transformation) → AX(AI Transformation)

산업혁명 비교표 — 2026 칸을 더하다

원본 110~138 쪽에서 한 칸씩 채워 가던 표입니다

	1 차	2 차	3 차	4 차	2026 지금
시기	1784	1870	1969	… 2020	2023~ 생성형 AI
에너지	석탄	석유	전기	재생에너지	AI 데이터센터 전력
소통	전신	전화	인터넷	사물인터넷	사람 ↔ AI 대화
운송	철도	자동차	스마트운송	로컬생산 (3D 프린터)	자율주행
영향력	소비재 · 경공업	전기화학 · 중화학	정보통신	모든 산업	모든 지식노동
패러다임	기계	대량생산	디지털	인공지능	AI 에이전트
키워드	증기	전기	통신 · 정보	무인 · 데이터	자동화된 판단

글로벌 시가총액 상위 5 개 기업의 변화

에너지 · 금융 → 플랫폼 → AI 인프라

순위	2007	2017	2026 (11 월)
1 위	페트로차이나	애플	애플
2 위	엑슨모빌	알파벳 (구글)	엔비디아
3 위	제너럴일렉트릭	마이크로소프트	마이크로소프트
4 위	차이나모바일	아마존	알파벳 (구글)
5 위	중국공상은행	페이스북	아마존

2026 년은 2026-11-02 종가 기준 (chartrow.com). AI 칩 회사가 세계 2 위가 되었습니다

> 디지털 트랜스포메이션 혁신은 모든 산업에 대하여 피할 수 없는 과정이며 , 변화에 성공한 기업만이 살아남을 수 있다 .

— 원본 141 쪽

2026 년 , 그 변화의 이름은 AI 트랜스포메이션입니다

원본 속 사례 , 그 후

원본 142~153 쪽에서 소개한 디지털 전환 사례들의 현재

사례	원본 당시 기대	2026 년 현재
IBM Watson	AI 의료 혁신	2022 년 Watson Health 매각 , watsonx 로 재편
adidas Speedfactory	로봇 공장 생산 귀환	2019 년 폐쇄 발표 , 기술은 아시아 협력사로
GE Predix	산업 IoT 플랫폼	야심 축소 , GE Vernova 소프트웨어 부문으로
Siemens MindSphere	산업 IoT 운영체제	Insights Hub 로 이름을 바꿔 Xcelerator 에 통합
NVIDIA	GPU 기반 AI 플랫폼	시가총액 세계 2 위의 AI 인프라 기업

교훈 : ' 플랫폼 ' 을 선언한 기업보다 고객의 문제를 푼 소프트웨어와 인프라가 살아남았다

소프트웨어가 제조를 먹고 , AI 가 소프트웨어를 쓴다

원본의 결론

소프트웨어 업체가 제조를 인수

- 제조를 돕는 IT → 제조를 이끄는 SW
- 플랫폼 · 클라우드 경쟁
- 개발자 확보가 곧 경쟁력

2026

AI 가 소프트웨어 개발을 바꿈

- 코드 작성 · 테스트 · 운영 분석까지
- 1 인 · 작은 팀도 제품을 만든다
- 도메인 지식 + AI 활용이 경쟁력

산업 AI 는 현장을 아는 작은 팀에게 가장 큰 기회입니다

04

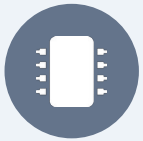
자원과 비용

가장 비싼 자원을 최적화하라 — 그 자원은 누구인가



자원 비용의 역전 , 그리고 한 번 더

원본 165~186 쪽 : 과거에는 컴퓨터가 , 지금은 사람이 더 비싸다



1955~

컴퓨터가 비쌌다

1955 년 4.5MB 코드 ,
1956 년 5MB HDD.
개발이 느려도 최소
자원으로 최적화 .



원본

사람이 더 비싸다

자원을 더 써도 빨리 개발 .
위챗은 10 명이 1 억 명
서비스 .



2026

토큰이라는 새 비용

사람 > AI 토큰 > 컴퓨터 .
1 인 + 에이전트로 서비스 ,
가장 비싼 건 사람의 판단 .

CPU 1 사이클을 1 초로 늘려 보면

원본 170~178 쪽의 비유에 마지막 두 줄을 더했습니다

작업	실제 시간 (대략)	1 사이클 = 1 초로 환산
CPU 1 사이클	0.3 ns	1 초
메모리 접근	약 100 ns	6 분
SSD 읽기	50~150 µs	2~6 일
HDD 읽기	1~10 ms	1~12 개월
샌프란시스코 → 뉴욕 왕복	40 ms	4 년
VM 재부팅	4 초	423 년
LLM API 응답 1 회	2~10 초	약 200~1,000 년
코딩 에이전트 작업 1 회	약 10 분	약 6 만 년

가장 비싼 자원 (YOU) 을 최적화하세요 .

— 원본 187 쪽

2026 년의 YOU 는 ' 시간 ' 보다 ' 판단과 주의력 ' 이 가장 비쌉니다

현재 개발 패러다임 — 원본 → 2026

원본 188~190 쪽의 세 가지를 다시 씁니다



188 쪽

언어

원본 : 가독성 좋고 생산적인 언어
2026: 자연어 명세 + 사람이 리뷰할 수 있는 언어



189 쪽

구조

원본 : MSA (팀 · 네트워크 기반)
2026: 모듈러 모놀리스 먼저 , 필요하면 MSA



190 쪽

운영

원본 : DevOps · NoOps 로 최소 인원
2026: DevOps + 에이전트로 1 인 운영까지

그래도 Python 인가 ?

GitHub Octoverse 2025: TypeScript 가 처음 1 위 , Python 은 2 위이자 신규 AI 저장소의 절반 가까이



배우기 쉬움

여전히

에이전트가 쓴 코드를 사람이 읽고 고치기 쉬움



생산성

강화

LLM SDK · 데이터 · ML 생태계의 중심



가독성

더 중요

리뷰가 곧 개발 — 읽기 쉬운 코드가 이긴다



커뮤니티

변화

웹 프런트는 TypeScript, AI · 백엔드는 Python 병행

05

DevOps 의 진화



개발 사이클이 끊임없이 도는 구조에 에이전트가 들어왔다

개발 방식의 진화

원본 199~201 쪽의 세 단계에 네 번째를 더했습니다



전통 방식

개발 → QA → 운영 , 부서별 순차 진행



TDD 개선

개발 → TDD → QA → 운영 , 테스트 먼저



DevOps

개발 · QA · 운영이 하나로 자동 순환



+ 에이전트

사이클 안에서 에이전트가 코드 · 테스트 · 분석 수행

TDD 는 다시 주목받습니다 — 테스트가 에이전트에게 주는 가장 정확한 지시서이기 때문입니다

혁신 게임의 규칙 — 무게중심이 이동했다

원본 202 쪽 : 성공은 세 질문의 교집합



Design

Desirable

Do you want this?
사용자가 원하는가 — 이제 가장 중요한 질문



Technology

Feasible

Can we do this?
AI 덕분에 ' 만들 수 있나 ' 의 답은 대부분 Yes



Business

Viable

Should we do this?
만들기 쉬워질수록 더 어려워진 질문

DevOps 단계별 — 원본 도구에서 2026 에이전트까지

원본 208~211 쪽 단계 그대로 — 단 , 배포 승인 · DB 마이그레이션 · 운영 키는 사람이 쥡니다

단계	원본	2026 도구	에이전트 역할
plan · code	이슈 트래커 · IDE	이슈 → 명세 (spec) 문서	명세 초안 , 코드 작성
build · test	1 일 n 회 빌드 , Selenium	CI(GitHub · Gitea Actions), Playwright	테스트 작성 , 실패 원인 분석
release · deploy	Vagrant · Ansible, 사이트별 릴리스	컨테이너 이미지 + GitOps	Dockerfile · 파이프라인 초안
operate · monitor	모니터링 → 버그 리포트 → plan	OpenTelemetry, 알림 → 이슈 자동 생성	로그 · 트레이스 분석 , 수정 PR 제안

DevOps 측정 — 2013 년 표에서 DORA 지표로

원본 213 쪽 : Amazon 은 하루 23,000 번 , 일반 기업은 9 개월에 한 번 배포 (The Phoenix Project, 2013)

DORA 지표	의미	AI 시대에 보는 이유
배포 빈도	얼마나 자주 배포하는가	에이전트로 변경량이 늘어도 작게 자주 배포하는지
변경 리드 타임	커밋에서 운영까지 걸리는 시간	리뷰 병목이 생기지 않았는지
변경 실패율	배포 후 장애로 이어진 비율	AI 생성 코드가 품질을 떨어뜨리지 않는지
복구 시간	장애에서 정상화까지 걸리는 시간	에이전트가 원인 분석을 앞당기는지

숫자로 보는 AI 와 DevOps

Google DORA 보고서 (2024 · 2025)

90%

개발자 AI 도입률

2025 년 . 2024 년 76% 에서 증가

−7.2%

배포 안정성

2024 년 : AI 도입 25% 증가당 안정성 변화 — 기본기 없이 쓰면 불안정

7 가지

효과를 본 조직의 관행

작은 배치 , 강한 버전 관리 , 좋은 내부 플랫폼 , 사용자 중심 등

2025 년은 측정 방식이 달라 직접 비교 불가 — 처리량 저하는 사라졌고 안정성 영향은 남음

> DevOps 는 고객의 요구에 대해 얼마나 빨리 응대하고 대응하는가를 알 수 있는 지표 .

— 원본 212 쪽

AI 는 속도를 올려 주는 증폭기 — 안정성은 여전히 DevOps 기본기가 지킵니다

06

클라우드와 컨테이너

Docker 개발 환경에서 에이전트 샌드박스까지



aaS 계층 — 원본에 더한 것들

원본 215~237 쪽 : SaaS · PaaS · IaaS · BaaS · CaaS

계층	원본 설명	2026 추가 · 변화
IaaS	VM · 스토리지 · 네트워크	GPU 클라우드 — AI 학습 · 추론용 자원
PaaS	런타임 · DB · 웹 서버	서버리스 함수 · 엣지 실행 환경
BaaS	백엔드 서비스	인증 · DB · 스토리지 일체형 — 바이브 코딩의 단골
CaaS	컨테이너 서비스	Kubernetes 가 사실상 표준 , 온프레미스는 k3s
SaaS	완성된 애플리케이션	대부분의 SaaS 에 AI 기능 내장
모델 API	(없음)	LLM 을 API 로 — 토큰 과금 , 온프레미스 LLM 도 선택지

12-Factor App, 에이전트 시대에 더 중요해진 4 가지

원본 220~226 쪽 — 12factor.net 의 원칙은 그대로 유효합니다



III. 설정

환경변수

API 키를 코드에 넣지 않기 — AI 생성 코드의 단골 실수



X. 환경 일치

개발 = 운영

devcontainer 로 사람 · 에이전트 · 운영이 같은 환경



XI. 로그

스트림

구조화된 로그 — 에이전트가 읽고 원인을 찾기 쉬움



IX. 폐기 가능성

쓰고 버림

에이전트 작업 컨테이너는 망가지면 지우고 새로

컨테이너의 진화

원본 227~236 쪽 : 실생활의 컨테이너처럼 표준 규격이 산업을 바꿨다



Docker

VM 보다 가벼운 격리 , 이미지로 배포



Compose · Swarm

여러 컨테이너를 정의하고 묶어 실행



Kubernetes

오케스트레이션의 사실상 표준



에이전트 샌드박스

에이전트가 마음껏 실행해도 호스트는 안전

원본의 멀티 클라우드 그림은 여전히 유효 — 이제 온프레미스 GPU 서버도 같은 컨테이너로 운영합니다

Pet & Cattle — 에이전트 작업 환경에도

Pet



이름을 붙여 키우는 서버

- Molly 같은 이름
- 손으로 하나하나 설정
- 아프면 간호해서 살린다

vs

Cattle



번호로 관리하는 서버

- vm0022 같은 번호
- 다른 것과 거의 동일
- 아프면 새것으로 교체

에이전트가 쓰는 개발 환경도 Cattle 로 — 망가지면 지우고 이미지에서 다시 만든다

Ready-made 개발 환경 — 원본 → 2026

원본 (Docker Hub)

mcchae/xfce-pyenv 이미지

- docker-compose 로 실행
- 컨테이너 안 XFCE 데스크톱 + PyCharm
- pyenv 로 파이썬 버전 관리

2026

devcontainer + 에이전트

- devcontainer.json 표준 규격
- IDE · 터미널이 컨테이너에 접속
- uv 로 파이썬 · 패키지 관리

' 준비된 개발 환경 ' 이라는 원본의 아이디어 그대로 — 이제 사람과 에이전트가 함께 씁니다

실습 — devcontainer 안에서 에이전트 돌리기

- **이미지로 환경 고정**
  팀원 · 에이전트 · CI 모두 같은 환경
- **uv 로 의존성 관리**
  pyenv + pip 대신 하나로
- **작업 폴더만 마운트**
  /workspace 밖은 못 건드림
- **망가지면 Rebuild**
  Cattle 원칙 그대로

```
// .devcontainer/devcontainer.json
{
  "name": "py-agent-dev",
  "build": { "dockerfile": "Dockerfile" },
  "workspaceFolder": "/workspace",
  "postCreateCommand": "uv sync"
}

# .devcontainer/Dockerfile
FROM python:3.12-slim
RUN apt-get update && apt-get install -y git tmux \
 && pip install uv
WORKDIR /workspace

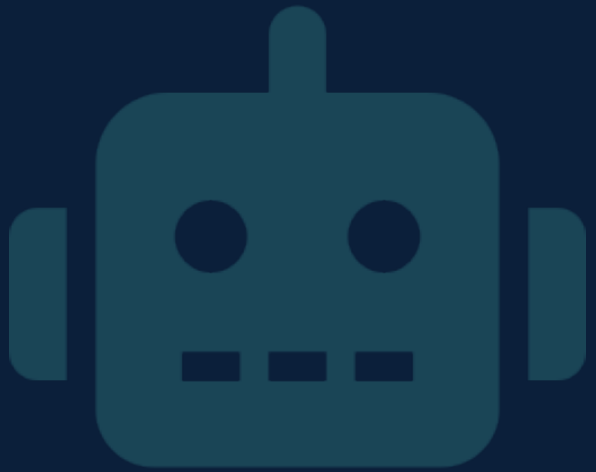
# inside the container
$ uv run pytest
$ claude          # or codex
```

컨테이너는 에이전트에게 '마음껏 해도 되는 방' — 망가져도 내 컴퓨터는 안전합니다

07

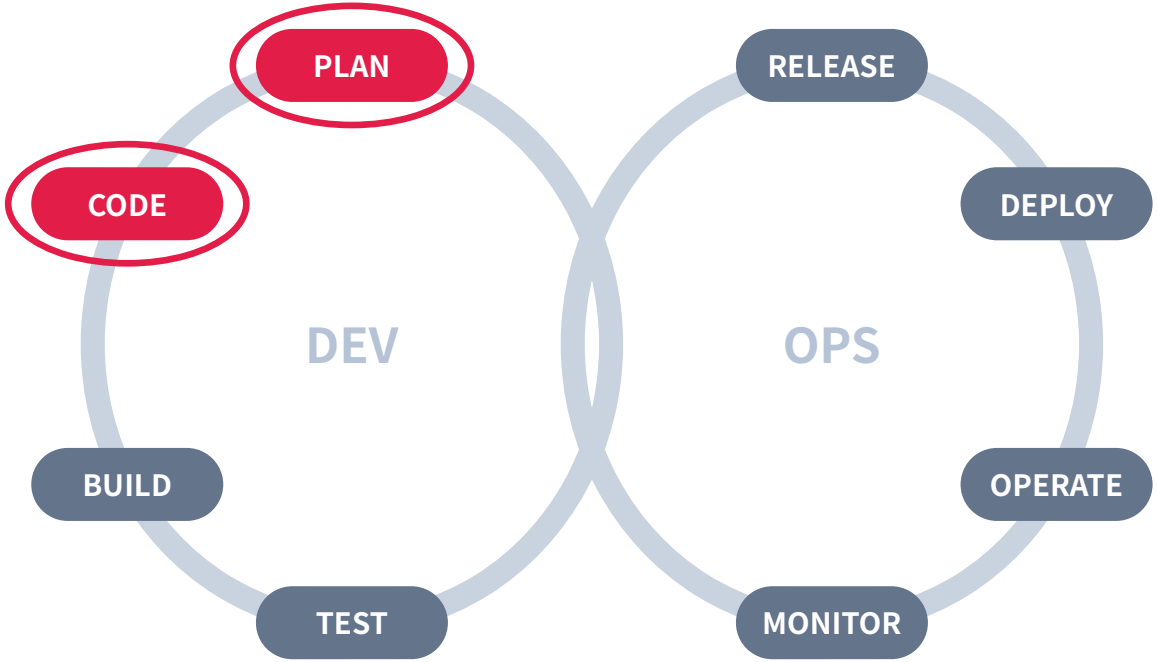
무인화 걸림돌 , 그 후

원본 241 · 274 쪽에서 동그라미 친 두 단계



원본의 무인화 걸림돌 : Code 와 Plan

빌드부터 모니터링까지는 자동화했지만 , 두 단계는 사람이 해야 했다



자동화 완료

build · test · release · deploy · operate · monitor

무인화 걸림돌

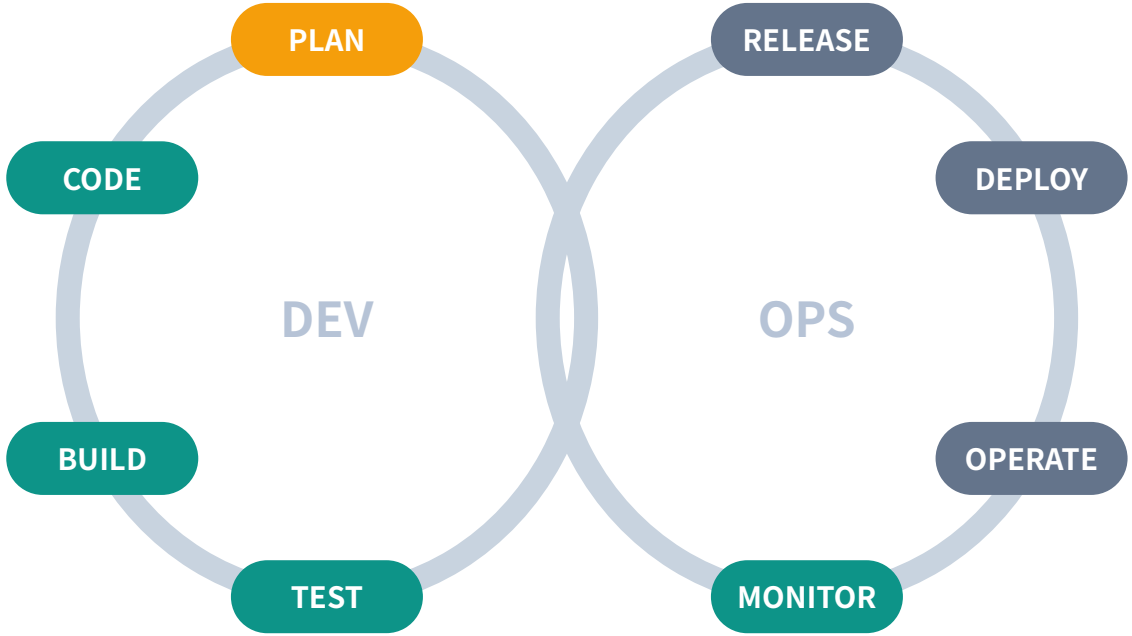
plan(무엇을 만들까) 과 code(어떻게 만들까) 는 사람의 몫

원본의 해법

개발 환경을 표준화 · 자동화해 도메인 로직 구현에 집중

2026: Code 는 에이전트에게 , Plan 은 여전히 사람에게

걸림돌 하나는 풀렸고 , 새로운 걸림돌이 생겼다



에이전트가 수행

code · build · test 반복 , 모니터링 로그 분석

여전히 사람

plan — 무엇을 왜 만드는가 , 그리고 배포 승인

새 걸림돌

명세의 품질과 리뷰 속도 — 생성 속도를 검증이 못 따라감

원본의 결론 세 가지 , 2026 버전

원본 275~277 쪽



275 쪽

개발 환경 표준화

원본 : Docker 이미지
2026: devcontainer + AGENTS.md 로 사람과 에이전트의 규칙까지 표준화



276 쪽

개발 환경 자동화

원본 : 빌드 · 테스트 · 배포 자동화
2026: CI + 에이전트 루프 , 테스트가 통과할 때까지 반복



277 쪽

무엇에 집중할까

원본 : 도메인 로직을 어떻게 구현할 것인가
2026: 무엇을 만들고 , 맞게 만들었는지 검증하는 것

08

마치며

변화하며 살아남는 것



원본 → 2026 한 장 요약

위는 원본의 개념 , 아래는 2026 년의 답



디지털 전환

AI 전환 (AX)



사람이 코딩

에이전트가 코딩



YOU 최적화

판단 · 주의력 최적화



DevOps

DevOps + 에이전트



Phoenix 지표

DORA 지표



Docker 개발 환경

devcontainer 샌드박스



pyenv · PyCharm

uv · 터미널 에이전트



블로그 기록

AGENTS.md · ADR

“

변화하며 살아남는 것은 기업의 문제 이전에 각자 개개인의 문제 .

— 원본 283 쪽

AI 시대에도 이 문장은 한 글자도 바꿀 필요가 없습니다

Change · Challenge · Chance

원본 284~286 쪽 , 2026 년의 의미



변화

Change

코드를 쓰는 주체가 바뀌었다 .
사람의 일은 명세와 검증으로 이동



도전

Challenge

도메인 이해 · 설계 · 리뷰 역량을 키워라 .
에이전트를 지휘하는 법을 익혀라



기회

Chance

1 인 · 작은 팀이 제품을 만드는 시대 .
현장을 아는 사람에게 가장 큰 기회

참고 자료

원본 자료와 2026 년 기준 자료

자료	주소
The Twelve-Factor App (원본 참고)	12factor.net/ko
DORA — State of AI-assisted Software Development	dora.dev
GitHub Octoverse 2025	github.blog/news-insights/octoverse
Development Containers 규격	containers.dev
uv — Python 패키지 · 프로젝트 관리	docs.astral.sh/uv
AGENTS.md — 에이전트 규칙 파일	agents.md
시가총액 순위 (2026-11-02 기준)	chartrow.com/lists/biggest-companies

실습 과제와 Q&A

내 Python 프로젝트 하나를 골라 2026 년식 개발 환경으로 옮겨 보세요



- devcontainer.json + Dockerfile 로 개발 환경 정의 , uv 로 의존성 관리
- 컨테이너 안에서 에이전트에게 기능 1 개와 테스트 구현 맡기기
- 회고 한 단락 : Plan 과 리뷰는 내가 , Code 는 에이전트가 — 어디까지 맡겼나