2기 3팀

Teracle 321
: AI 기반 UI 테스트 자동화

AI 기반 UI 테스트 자동화

소스코드를 분석하여 사용자 행동 흐름을 파악하고, Playwright로 실행 가능한 E2E 테스트를 자동으로 생성하는 AI 시스템

이해관계자 |

- 개발자, IT 프로젝트 수행 조직
- 주제 발의조직 : AI ITS 혁신팀

요구사항 |

- 소스코드를 분석하여 사용자 행동 흐름을 분석하고,
  테스트 시나리오를 자동으로 생성한 후 Playwright 등
  UI 테스트 도구를 활용하여 실행 및 리포트화

Pain Point |

- 테스트 시나리오 작성에 드는 시간과 비용
- 테스트 품질 일관성 확보
- 테스트 결과 공유 및 추적의 불편함

문제 정의: QA 자동화가 실패하는 세 가지 구조적 한계

테스트 유지보수 비용 과부하

→ 예측 불가능한 비용으로 인한 경영 리스크 증폭

**TCO(Total Cost of Ownership)*
시스템을 도입해 운영·유지보수까지 포함한 전체 생애주기 비용을 의미
초기 구축비용뿐 아니라 인력, 유지보수, 장애 대응, 인프라 비용까지 모두 고려

코드/UI 변경 시 테스트의 50% 이상이 재작성 필요
PR당 평균 2시간 이상의 고급 인력 시간이 소모
테스트 문서·케이스 수정 비용이 누적되어 TCO 증가

릴리즈 병목 현상 TTM 지연

→ 순차 실행에 의존하며 TTM 경쟁력 상실 및 자원 최적화 실패

작은 변경에도 전체 테스트 재실행
영향도 분석 부재 CI/CD 실행 시간 증가
PR 단위 피드백 지연 릴리즈 결정이 느려짐

품질 신뢰도 저하 의사결정 리스크 증가

→ 테스터 지식 사일로 및 전문성 편차가 일관성 없는 품질을 초래함

**Flaky Test*
같은 코드인데도 실행할 때마다 성공/실패가 랜덤하게 바뀌는 불안정한 테스트 의미
환경·타이밍·비결정성 때문에 자동화 품질을 떨어지게함

담당자별 편차로 Flaky 테스트 증가
테스트 커버리지·증적이 표준화되지 않아 추적 어려움
품질 신뢰도 저하 릴리즈 의사결정 리스크 증가

[출처]
1. https://idealink.tech/blog/understanding-software-testing-costs-development-breakdown
2.https://www.reddit.com/r/devops/comments/1op2qri/reduce_ci_cd_pipeline_time_strategies_that/
3.https://www.virtuosoqa.com/post/self-healing-continuous-testing

문제 정의: 기존 자동화 도구들의 공통적인 한계

기존 자동화의 구조적 한계

높은 비용 & 낮은 효율

수동 정의 기반 비용 과다

높은 진입 장벽 (개발자 역량 의존)

단순 매칭 취약 (Auto-Healing 실패)

영향도 분석 부제 (전수 테스트 강요)

구조적 한계의 핵심 원인 분석

근본적인 자동화 구조의 문제

수동 정의 의존 (사람의 개입 필수)

지능화/자동화 미흡 (AI 부재)

유지보수 비용 폭증

Teracle의 구조적 한계 극복 솔루션

자동화 혁신 3대 전략

시나리오 자동 생성

Hybrid Auto-Healing

정밀 영향 분석

AI 기반 지능형 자동화 구현 / 비용 획기적 절감 / 신뢰도 및 완성도 확보

QAOps를 실현하는 AI 플랫폼:
요구사항 업로드 → 자동 테스트 생성→코드 변경 시 자가 복구로 지속적 품질 보증

QA의 패러다임 전환: QAOps 실현

타겟 사용자:
- PM : 요구사항 → 테스트 자동화 변환
- 개발자 : PR마다 자동 검증, 스크립트 유지보수 최소화
- QA Engineer : 수동 반복 테스트 → 전략적 품질 관리

가치 제안: 핵심 기능

완전 자동 생성

테스트 작성 비용 Zero
AI 자율 도출

요구사항 기반 시나리오·코드 자동 생성
→ 초기 TCO 99% 절감

PDF 업로드 → AI 분석 → 실행 가능한 코드

Hybrid Self-Healing

테스트 유지보수 비용 Zero
2단계 Hybrid 자가 복구

휴리스틱 + 의미기반 복구로
85% 자동 Healing
→ 유지보수 비용 Zero화

1. 빠른 경로: 휴리스틱 (70% 케이스, 2초)
2. 정밀 경로: 608차원 Vector DB (30% 케이스)
3. Confidence Score로 오탐 방지 (임계값 0.75)

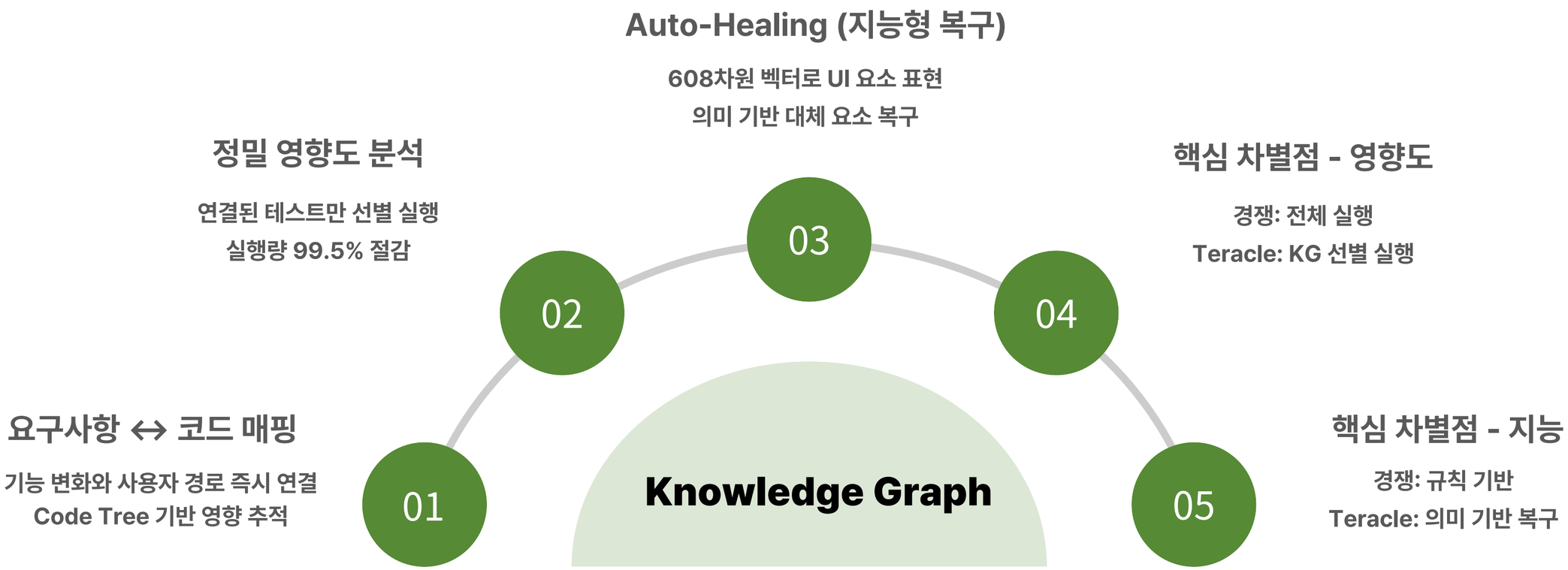
정밀 영향 분석

CI/CD 자원/시간 Zero
Knowledge Graph 기반
선별 실행

변경 영향 테스트만 선별 실행 (99.5% 절감)
→ CI/CD 속도 극대화

Knowledge Graph 기반 영향 분석
→ 변경된 코드 영향받는 테스트만 자동 선택

Knowledge Graph는 자동화를 넘어,
정밀 영향 분석·요구사항/코드 매핑·의미 기반 복구를 수행하는
Teracle의 지능형 QAOps 엔진입니다.



UI 변경에도 테스트 스크립트를 자동 복구하는 Teracle의 지능형 안정화 엔진

UI 변경에도 테스트 스크립트를 자동 복구하는 Teracle의 지능형 안정화 엔진

608차원 Selector Feature Vector

시각적 특징 : 256차원 +텍스트 의미 : 128차원 + DOM 구조 : 128차원 + 속성 : 64차원 + 위치/상태 : 32차원 = 608차원

기능 설명: Auto-Healing 엔진

벡터화 & 저장 |

DOM 요소 분석

608차원 벡터 변환

Vector DB 저장

3단계 복구 프로세스|

01 휴리스틱 기반 1차 복구

- 간단 문제 / Timeout 해결
- 선택자 인덱스 조정 (동적 리스트)

02 Vector DB 기반 2차 복구

- 608차원 공간 검색 : 동일 의미 요소 자동 발견

03 Confidence Score 기반 오탐 방지

- 신뢰도 점수 계산 (0.0 ~ 1.0)
- 임계값 미만 → HITL 요청

핵심 목표 수치 |

85%+
자동 복구 성공률

70%↓
유지보수 시간 절감

40%↓
flaky Test 감소

99.5%+
테스트 실행 안정성

BEFORE

100개 실패
→ 100개 수동 조정

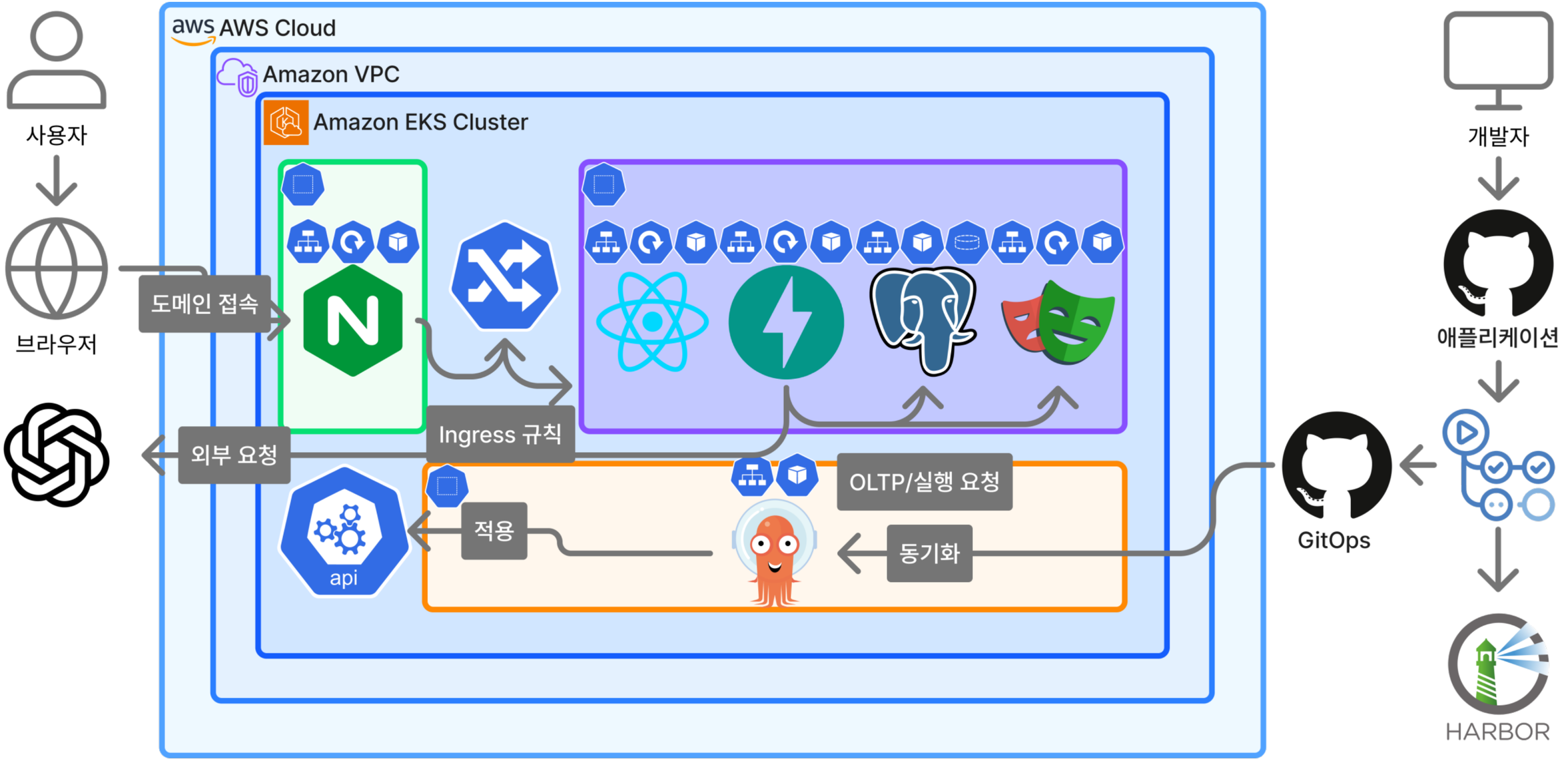
2-3일 소요

AFTER

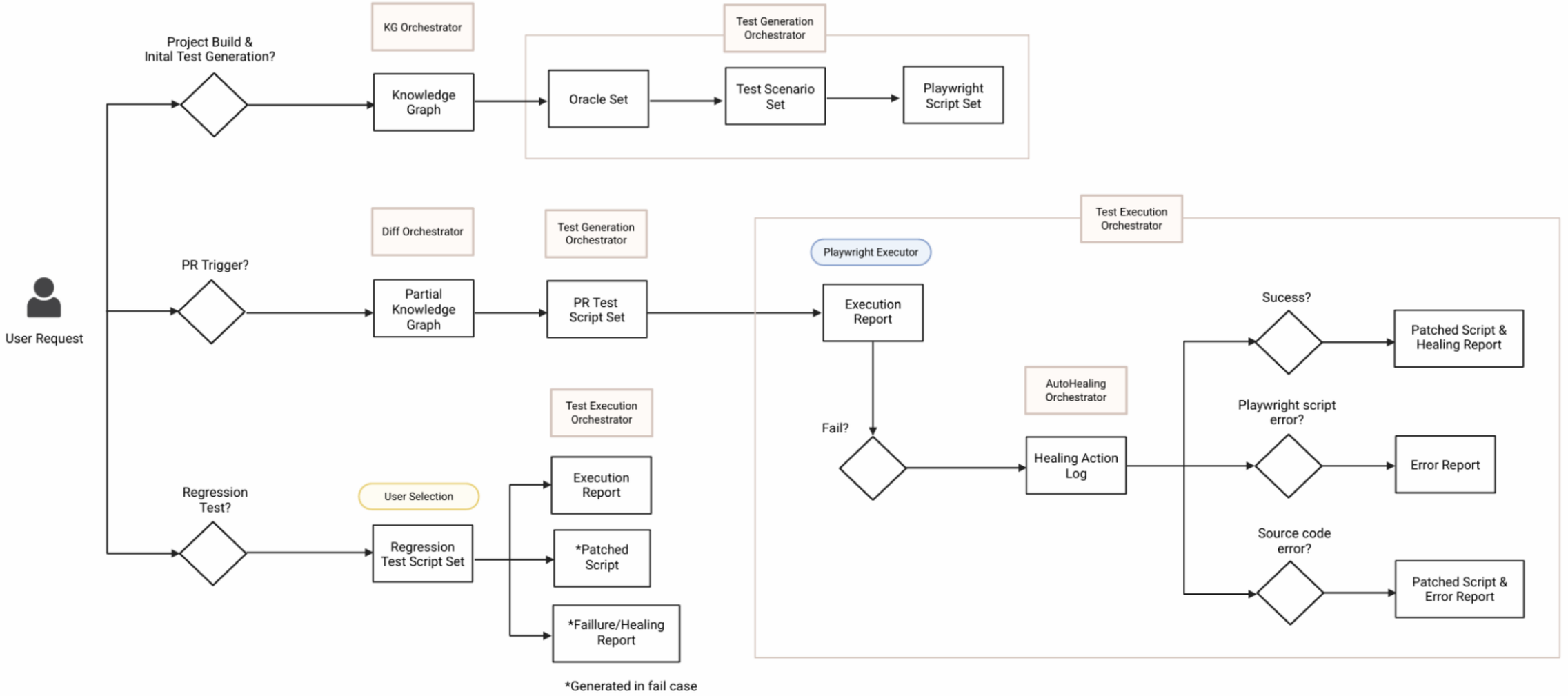
85개 자동 복구
→ 15개만 수정

반나절 소요
(70% 시간 절감)

시스템 구조: Deployment/Infra Architecture

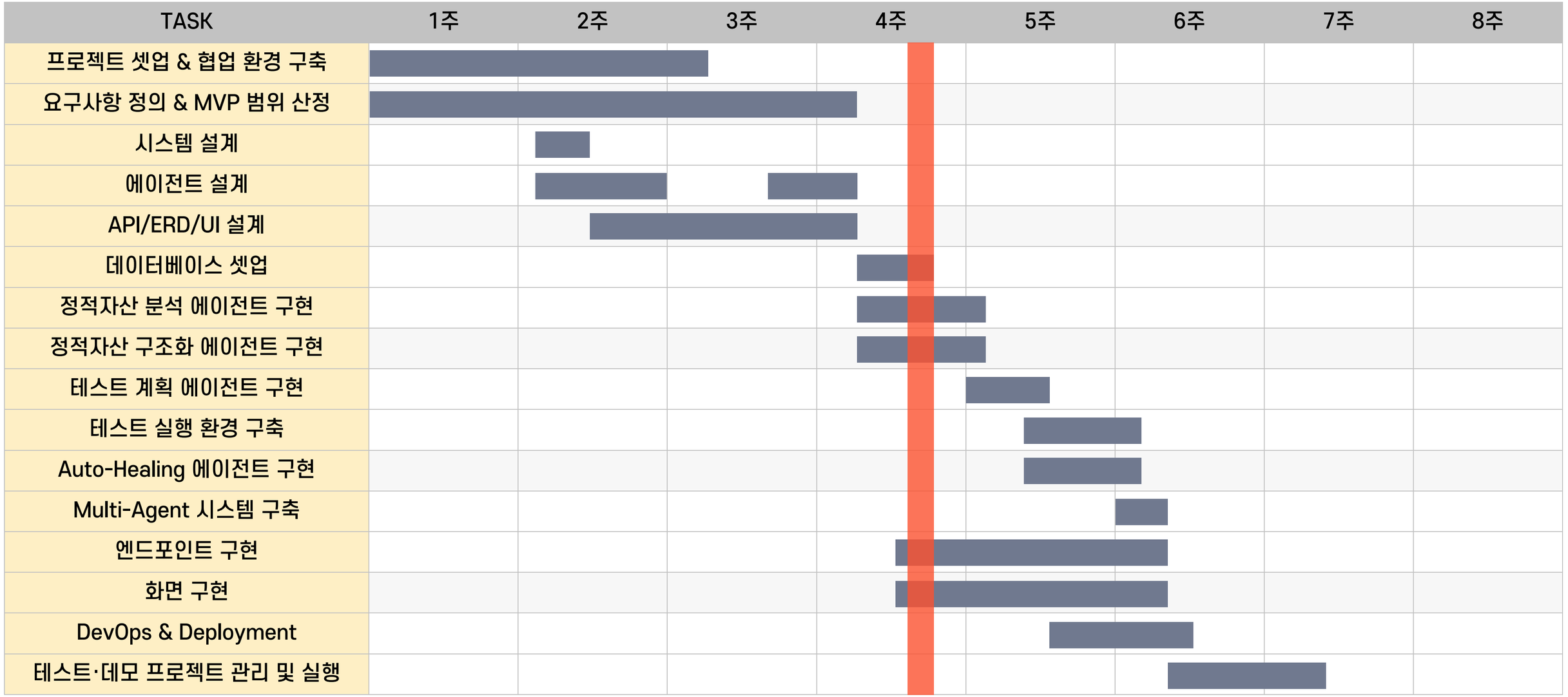


시스템 구조: AI Agent Orchestration Flow



로드맵: 간소화된 간트 차트

2025.11.28



로드맵: 향후 개발 계획

1

Week 1

목표:

분석·매핑 계층과
실행 파이프라인의 뼈대 구축

핵심 활동:

FastAPI, DB, Vector DB,
Runner 등 기반 인프라 세팅

요구사항·API·DB·소스코드
분석/구조화 에이전트 1차 구현

Req/Code Analyzer →
Knowledge Graph 스키마 &
매핑 파이프라인 구성

2

Week 2

목표:

테스트 생성·실행 엔진의
MVP 완성

핵심 활동:

Scenario / Oracle /
Test Case / Script
Synthesizer 에이전트 구현
(HITL 포함)

Playwright 기반 테스트 스크립트
생성·실행 Executor 연결

React 기반 최소 UI에서
프로젝트 선택 → 테스트 실행까지
E2E 흐름 완성

3

Week 3

목표:

Auto-Healing 고도화 및
배포·데모 안정화

핵심 활동:

규칙 + Vector DB를 조합
Hybrid Auto-Healing 구현

실패/Flaky 테스트 자동 감지
재시도 및 Healing 신뢰 지표 산출

Docker/Kubernetes + CI/CD
배포 환경 정리 및 데모 프로젝트
통합 테스트/발표 준비

마무리: 팀원 소개

원기훈
역할 : PM, FE

정광진
역할 : BE, CI/CD

윤소현
역할 : BE

김선형
역할 : UI/UX, BE

한창현
역할 : BE

송지윤
역할 : BE

Teracle: Test + Oracle → Test + Miracle! 테스트의 기적을 선도하는 여정을 시작합니다.

감사합니다