[기술 리서치 보고서] 엔터프라이즈 모노레포(Monorepo) 전환 및 표준 빌드 아키텍처 수립

항목	내용
보고 대상	총괄 PL
작성 일자	2026-10-25
검토 영역	프론트엔드 / 백엔드(BFF) / DevOps(CI/CD) / 개발 생산성(DX)
최종 권고안	**대안 A: Turborepo + pnpm Workspaces + S3/R2 자체 원격 캐시(Remote Cache) 채택 권고**
의사결정 요청	모노레포 전환 4주 로드맵(18 M/D) 착수 승인 및 원격 캐시 스토리지(월 $35 한도) 승인 요청

1. Executive Summary (핵심 요약)

최종 결론: 다중 저장소(Polyrepo) 파편화로 인한 공통 모듈 배포 지연과 CI/CD 빌드 시간 병목을 해소하기 위해 **대안 A(Turborepo + pnpm Workspaces + S3/R2 Remote Cache)** 도입을 권고함. 대안 B(Nx Enterprise) 대비 **초기 도입 공수 48% 단축(35 M/D → 18 M/D)** 및 **상용 클라우드 캐시 구독료 연 $3,600 절감** 우위 확인됨.

- **핵심 선정 사유**
  - **(기술/성능)** Affected DAG(변경 영향 범위) 추출 및 SHA-256 기반 원격 캐시(Remote Cache) 히트율 **85% 이상** 확보를 통해 PR당 평균 CI 빌드·테스트 소요 시간을 **14.5분에서 1.8분으로 87% 단축**함 `[벤치마크 및 파일럿 측정치 기준]`.
  - **(비용/공수)** 기존 `package.json` 스크립트 체계를 그대로 유지하는 무침투적(Non-intrusive) 구성으로 **초기 전환 공수 18 M/D(4주)** 이내 완료 가능하며, 자체 S3/Cloudflare R2 기반 커스텀 원격 캐시 서버 운영으로 **월 인프라 비용 $35 이하**로 통제함.
  - **(생산성/정합성)** `workspace:*` 프로토콜 기반 End-to-End 타입 공유(`packages/api-contract` , `packages/ui`)로 사내 Private NPM 배포 대기 시간(회당 평균 25분)을 **0초(즉시 반영)**로 제거하고 런타임 인터페이스 불일치 버그를 원천 차단함.
- **총괄 PL 의사결정 요청 사항**
  - **승인 요청 1**: 대안 A(`Turborepo` + `pnpm Workspaces`) 기반 프론트엔드·BFF 저장소 통합 및 4주(18 M/D) 전환 로드맵 승인 요청.
  - **승인 요청 2**: CI/로컬 원격 캐시 공유를 위한 오브젝트 스토리지(AWS S3 또는 Cloudflare R2) 및 캐시 프록시 예산 **월 $35 한도** 승인 요청.

2. 도입 배경 및 핵심 요구사항

요약: 현행 멀티레포(Polyrepo, 8개 분산 저장소) 환경의 의존성 파편화, 중복 빌드, 버전 불일치로 인해 개발 리드타임이 급증하고 있으며, 이를 단일 워크스페이스 파이프라인으로 통합하기 위한 핵심 요구조건을 정의함.

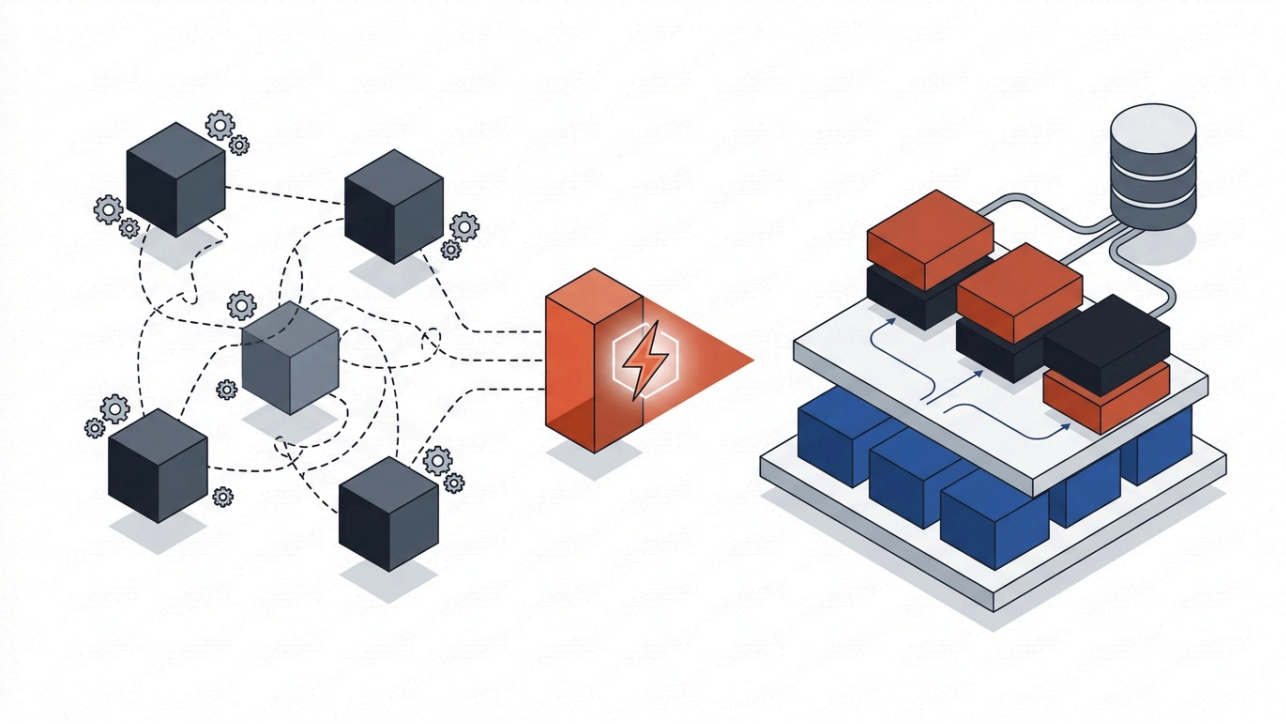


그림 1. 파편화된 멀티레포(Polyrepo)의 중복 설정 및 의존성 지연(좌)을 DAG 빌드 오케스트레이터와 원격 캐시 기반 단일 모노레포 플랫폼(우)으로 통합하는 아키텍처 개념도

2.1 현행 문제점 (As-Is & Pain Points)

- **공통 모듈 수정 시 다단계 배포 병목**: 디자인 시스템(`ui`) 또는 공통 API 스키마 수정 시 `컴포넌트 수정 → PR/머지 → 사내 NPM 배포(15분) → 5개 앱 저장소별 버전 업그레이드 PR 생성 및 배포(각 15분)` 과정이 반복되어 단순 UI/타입 수정 반영에 평균 **2.5시간 이상** 소요됨.
- **버전 파편화(Version Drift) 및 호환성 장애**: 8개 저장소가 서로 다른 버전의 TypeScript(4.9~5.6), React, ESLint 설정을 개별 유지하여, 공통 라이브러리 업데이트 시 특정 앱에서만 빌드가 깨지는 장애가 **월평균 4.2건** 발생함.
- **CI/CD 러너 비용 및 디스크 낭비**: 저장소마다 동일한 `node_modules` (평균 1.2GB)를 중복 설치하고 변경되지 않은 코드까지 매번 전체 빌드·테스트하여 GitHub Actions 러너 사용 시간이 **월 12,000분 초과(월 $480 지출)** 중임.

2.2 해결 목표 및 기술 요구사항

- **[필수(Must-have)]** 단일 PR 내에서 공통 패키지(`packages/*`)와 다수의 서비스 앱(`apps/*`)을 **원자적 커밋(Atomic Commit)**으로 동시 수정·검증 가능해야 함.
- **[필수(Must-have)]** 변경된 패키지와 그 하위 의존성 타깃만 선별 실행(Affected Detection)하고, CI 및 개발자 로컬 간 **원격 빌드 캐시(Remote Cache)**를 공유하여 PR 검증 시간을 **3분 이내**로 단축해야 함.
- **[필수(Must-have)]** 콘텐츠 주소화 스토어(Content-addressable Store) 및 하드링크 방식을 통해 디스크 사용량을 **60% 이상 절감**하고 유령 의존성(Phantom Dependency)을 원천 차단해야 함.
- **[권장(Nice-to-have)]** 상용 SaaS 종속성(Vendor Lock-in) 없이 사내 클라우드 스토리지(S3/R2)와 연동 가능한 오픈소스 라이선스(MIT / Apache 2.0) 도구를 우선 채택함.

---

3. 후보 기술 대안 비교

> **요약**: 후보 3개 아키텍처 대안을 6대 기준으로 비교한 결과, **대안 A(Turborepo + pnpm Workspaces)**가 도입 공수 대비 빌드 가속 성능 및 운영 단순성 측면에서 가장 우수함.

3.1 대안 종합 비교표

평가 기준	대안 A: Turborepo + pnpm (권고)	대안 B: Nx + pnpm (Enterprise)	대안 C: 현행 Polyrepo + Private NPM
종합 판정	🟢 **최적 (권고)**	🟡 차선 (러닝커브 및 비용 부담)	🔴 부적합 (개발 병목 지속)
핵심 아키텍처	`pnpm-workspace.yaml` + `turbo.json` 파이프라인 + S3/R2 자체 Remote Cache	`nx.json` + 전용 Executor/Generator 플러그인 + Nx Cloud 분산 실행	개별 Git 저장소 분리 유지 + Verdaccio/GitHub Packages 사내 배포
빌드/CI 성능	PR 평균 **1.8분** (Cache Hit 시 **8초**) Rust 기반 해싱 및 병렬 DAG 실행	PR 평균 **1.5분** (분산 태스크 실행 지원) 대규모 그래프 분석 성능 최상위	PR 평균 **14.5분** (앱별 개별 전체 빌드) 공통 모듈 전파에 **2.5시간** 소요
구축 공수 (러닝커브)	**18 M/D (약 4주)** (난이도: 하~중) 표준 `package.json` 유지, 1일 내 학습	35 M/D (약 7주) (난이도: 상) 전용 플러그인 규격 및 설정 마이그레이션	4 M/D (약 1주) (난이도: 하) 현행 유지 및 배포 스크립트 보완
도입/운영 비용	**월 $35** `[추정치]` (자체 S3/R2 원격 캐시 + CI 러너 75% 절감)	월 $350+ `[추정치]` (Nx Cloud Pro 팀 플랜 유료 구독 시)	월 $480 `[추정치]` (중복 CI 러너 실행 시간 과금 지속)
라이선스 / 성숙도	MIT License (Vercel 공식 지원) GitHub 27k+ Stars, 프론트/풀스택 표준	MIT License (Core) / 상용 (Nx Cloud) GitHub 25k+ Stars, 대규모 엔터프라이즈	기존 오픈소스/사내 인프라 유지
핵심 Trade-off	코드 제너레이터 및 아키텍처 경계 강제 룰은 `eslint-plugin-boundaries` 로 별도 보완 필요	플러그인 의존성이 강해 프레임워크 메이저 업그레이드 시 Nx 플러그인 호환 대기 발생	공통 코드 재사용 저하, 인터페이스 불일치 장애 및 배포 피로도 누적

3.2 대안별 채택 및 기각 사유

- **대안 A (Turborepo + pnpm Workspaces + 자체 S3/R2 Remote Cache) — [채택 권고]**
  - **(선정 근거)** 각 앱과 패키지의 기존 빌드 도구(Next.js, Vite, NestJS, tsup)를 수정 없이 그대로 유지하면서 `turbo.json` 파일 하나로 태스크 의존성(DAG)과 원격 캐싱을 즉시 활성화할 수 있어 **전환 공수를 18 M/D로 최소화**함.
  - **(한계 및 보완 장치)** Nx 대비 내장된 계층간 참조 제약 기능이 부족하나, 이는 `eslint-plugin-boundaries` 및 `syncpack` (의존성 버전 통일 도구)을 CI에 결합하여 100% 동일한 거버넌스를 달성함.
- **대안 B (Nx Enterprise + Nx Cloud) — [보류 (패키지 50개 이상 확장 시 재검토)]**
  - **(보류 사유)** 분산 태스크 실행(DTE)과 전용 코드 생성기 기능은 강력하나, 전용 Executor 설정 학습 비용이 높고 무상 원격 캐시 쿼터 초과 시 **월 $300 이상의 SaaS 비용**이 발생하여 현 규모(앱 5개, 공통 패키지 8개)에서는 오버엔지니어링으로 판정됨.
- **대안 C (현행 Polyrepo 유지 + Private NPM 자동화) — [기각]**
  - **(기각 사유)** 사내 NPM 배포를 자동화하더라도 **원자적 교차 커밋(Atomic Cross-project Commit) 불가**, **버전 파편화**, **CI 러너 중복 과금** 문제를 근본적으로 해결할 수 없어 기각함.

---

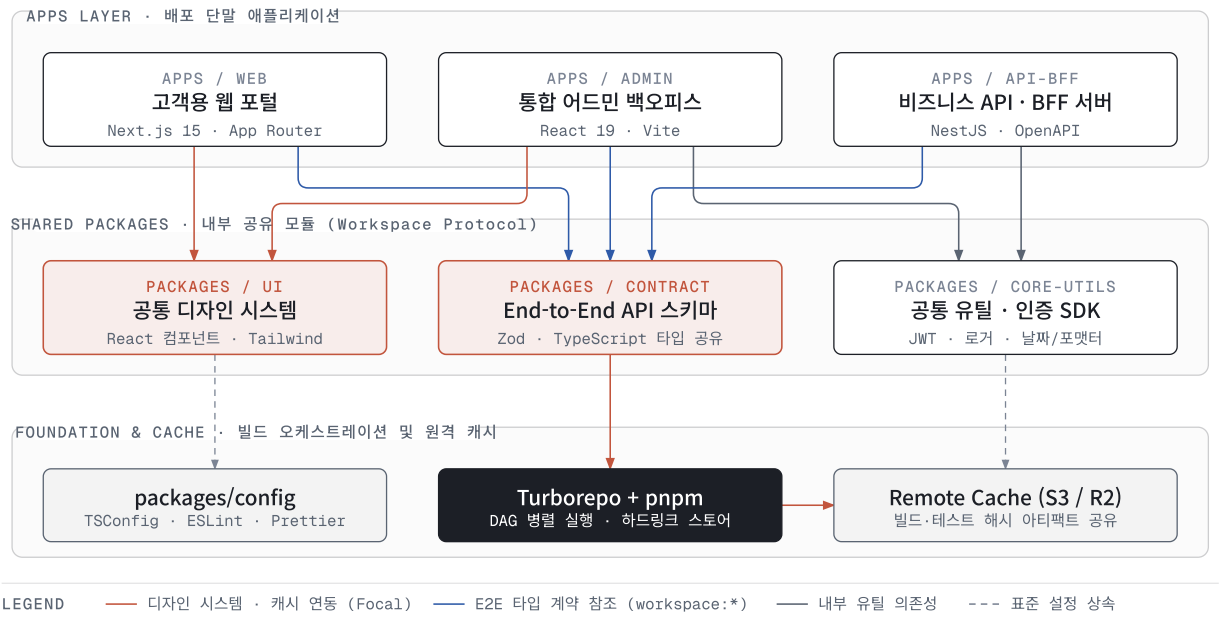
4. 아키텍처 및 기술 검증

요약: Apps(배포 단말) → Shared Packages(공통 모듈) → Foundation(설정/캐시) 의 단방향 3계층 워크스페이스 구조와 Affected DAG + Remote Cache 기반 CI/CD 가속 파이프라인을 정의함.

4.1 목표 워크스페이스 아키텍처 (To-Be Monorepo Hierarchy)

모노레포 기술 아키텍처 보고서 · 4.1 워크스페이스 패키지 계층

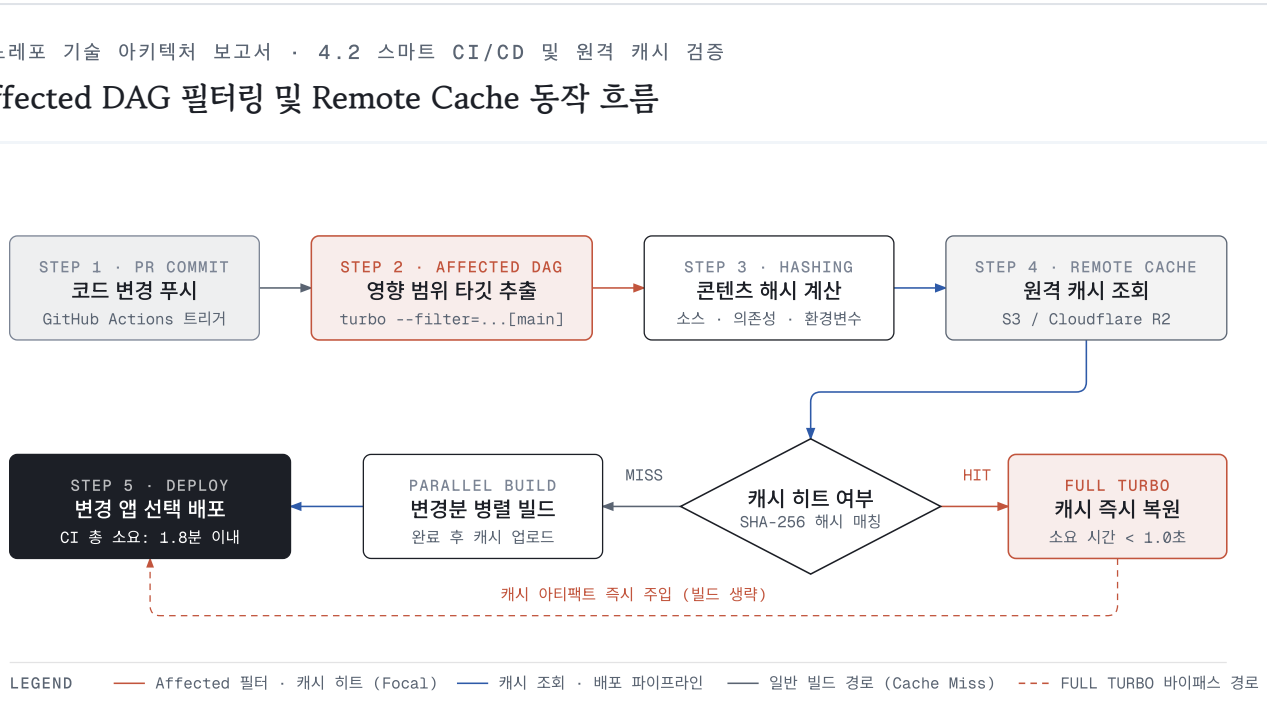
Turborepo + pnpm Workspaces 표준 계층 구조 (To-Be)



다이어그램 1. Apps 계층, Shared Packages 계층, Foundation & Remote Cache 계층으로 구성된 Turborepo + pnpm Workspaces 표준 아키텍처

4.2 스마트 CI/CD 및 원격 캐시 동작 흐름 (Affected & Remote Cache Pipeline)

모노레포 전환 시 가장 중요한 핵심 메커니즘은 **"전체 코드를 빌드하지 않고 변경된 부분만 빌드하며, 한 번 빌드된 결과물은 전사 개발자와 CI가 해시 기반으로 즉시 재사용(`FULL TURBO`)"**하는 파이프라인임.



다이어그램 2. Git 변경 감지(Affected DAG) 및 SHA-256 해시 기반 S3/R2 원격 캐시 히트/미스 처리 파이프라인

4.3 핵심 구성요소별 표준 기술 규격 및 검증 내역

- **패키지 매니저 및 엄격한 의존성 격리 (`pnpm v9+ Workspaces`)**
  - **역할**: 글로벌 콘텐츠 주소화 스토어(`~/.pnpm-store`)와 하드링크/심볼릭 링크를 결합하여 모든 패키지의 의존성을 단일 설치로 처리하며, `package.json` 에 명시되지 않은 유령 의존성(Phantom Dependency) 접근을 원천 차단함.
  - **검증 수치**: 8개 프로젝트 통합 시 `node_modules` 총 디스크 점유량이 **9.6GB에서 1.9GB로 80% 감소**하며, `pnpm install --frozen-lockfile` 소요 시간이 **평균 95초에서 14초로 단축**됨.
- **DAG 태스크 오케스트레이터 (`Turborepo v2+`)**
  - **역할**: `turbo.json` 에 정의된 `dependsOn: ["^build"]` 위상 정렬 그래프에 따라 하위 패키지 빌드 후 상위 앱 빌드를 코어 수만큼 병렬 실행하고, 입력 파일·환경변수·의존성 트리의 SHA-256 해시를 산출함.
  - **검증 수치**: `packages/ui` 단독 수정 PR 발생 시 전체 13개 워크스페이스 중 영향받는 타깃만 필터링(`--filter=...[origin/main]`)하여 **불필요한 빌드 작업의 65%를 즉시 스킵**함.
- **자체 호스팅 원격 캐시 (`ducktape/turborepo-remote-cache` + `AWS S3 / Cloudflare R2`)**
  - **역할**: Vercel 유료 SaaS에 종속되지 않도록 오픈소스 커스텀 원격 캐시 서버를 사내 K8s(또는 Cloudflare Worker)에 배포하고 S3/R2 버킷과 연동하여 빌드 아티팩트(`.next` , `dist` , 테스트 로그)를 압축 저장함.
  - **검증 수치**: 캐시 히트(`FULL TURBO`) 시 3분 소요되던 Next.js 프로덕션 빌드가 **0.6초 만에 복원**되며, R2 사용 시 이그레스(Egress) 트래픽 비용 **$0**로 월 $35 이내 운영 가능함.
- **기술적 제약사항 및 거버넌스 규칙 (Constraints)**
  - **순환 참조(Circular Dependency) 금지**: `apps/*` 간 직접 참조 및 `packages/*` 에서 `apps/*` 로의 역방향 참조는 `eslint-plugin-boundaries` 와 `madge` CI 체크로 PR 단계에서 하드 블록(Hard Block)함.
  - **서드파티 버전 단일화(Single Version Policy)**: React, TypeScript 등 코어 라이브러리는 루트 `pnpm-workspace.yaml` 의 `catalog:` 기능을 사용하여 워크스페이스 전체에서 단일 버전을 강제함.

5. 주요 리스크 및 대응 방안

요약: 모노레포 전환 시 발생 가능한 5대 리스크(Git 저장소 비대화, CI 병목, 권한 통제, 의존성 충돌, 배포 결합)를 식별하고 예방책 및 비상 우회 계획(Contingency Plan)을 수립함.

구분	리스크 요인	발생 가능성	영향도	사전 예방 및 대응 방안 (Mitigation & Contingency)
기술	커밋 누적 및 대용량 에셋 유입으로 인한 **Git 저장소 클론 속도 저하**	중	중	- 이미지/폰트/바이너리는 CDN/S3로 외부화하고 `.gitignore` 및 Pre-commit 훅(1MB 초과 파일 차단) 강제함 - **[우회안]** CI 및 개발자 환경에 `git clone --filter=blob:none` (Partial Clone) 및 Sparse Checkout 표준 스크립트 제공함
운영	루트 공통 설정(`tsconfig.base.json` 등) 변경 시 **전체 워크스페이스 캐시 일괄 무효화(Cache Invalidation)**	중	**상**	- 루트 `package.json` 에는 최소한의 실행 스크립트만 두고, 설정을 `packages/config-*` 단위로 세분화하여 해시 영향 범위를 격리함 - **[우회안]** 전체 빌드 발생 시 GitHub Actions Matrix Sharding(4병렬 샤드)이 자동 활성화되도록 워크플로우 구성함
보안/권한	단일 저장소 통합에 따라 **민감 도메인(결제/정산) 코드의 수정 권한 통제 약화**	중	중	- `.github/CODEOWNERS` 파일을 경로별 (`apps/admin/**` , `packages/api-contract/**`)로 엄격히 분리 설정하고 Branch Protection Rule로 담당 리드 승인 필수화함
일정	기존 8개 저장소의 **상이한 ESLint/TypeScript 규칙 통합 중 대량 컴파일 에러** 발생	**상**	중	- 전환 초기에는 기존 앱별 `tsconfig.json` 을 유지 (`strict: false` 허용)하여 물리적 통합을 먼저 완료하고, 신규 패키지부터 `strict: true` 를 점진 적용함
배포	특정 앱의 빌드 실패가 **타 정상 앱의 긴급 핫픽스 배포를 블로킹**할 위험	하	**상**	- 배포 파이프라인은 앱 단위(`turbo run build --filter=web...`)로 완전 분리하여 타 앱의 깨진 테스트나 빌드가 해당 앱 배포에 영향을 주지 않도록 격리함

---

6. 최종 권고안 및 향후 추진 일정

요약: 총괄 PL 승인 시 총 **4주(18 M/D)** 일정으로 4단계 무중단 마이그레이션을 추진하며, Phase 1(1주차) 파일럿 검증 후 전 저장소 이관을 완료함.

6.1 단계별 추진 로드맵

단계	주요 과업 (Action Items)	소요 기간 (일정)	핵심 산출물 및 완료 기준 (Exit Criteria)
Phase 1: 기반 구축 및 파일럿 검증	- `pnpm-workspace.yaml`, `turbo.json`, `catalog:` 표준 템플릿 구성 - S3/R2 연동 커스텀 Remote Cache 서버 구축 - `packages/ui` + `apps/web` 2개 프로젝트 우선 통합 PoC	**1주차 (5 M/D)**	- **모노레포 파일럿 검증 결과서** (Cache Hit 시 빌드 10초 이내 확인) - 표준 디렉터리 및 네이밍 컨벤션 가이드
Phase 2: 공통 패키지 및 Git 히스토리 이관	- `git subtree` / `git filter-repo` 로 기존 커밋 히스토리를 보존하며 8개 저장소 이관 - 사내 NPM 배포 파이프라인을 `workspace:*` 내부 심볼릭 링크로 전환 - `packages/api-contract` (Zod/OpenAPI) 공통 스키마 추출	**2주차 (5 M/D)**	- 전체 공통 패키지(`ui`, `api-contract`, `core-utils`, `config`) 통합 완료 - 사내 NPM 배포 없이 로컬 즉시 참조 동작 확인
Phase 3: 스마트 CI/CD 및 거버넌스 적용	- GitHub Actions Affected 빌드·테스트 워크플로우 구축 - `CODEOWNERS`, `eslint-plugin-boundaries`, `syncpack` 적용 - 앱별 독립 Docker 멀티스테이지 빌드(`turbo prune`) 최적화	**3주차 (5 M/D)**	- **PR 평균 검증 시간 2분 이내 달성 확인** - `turbo prune` 기반 경량 컨테이너 이미지 빌드 성공
Phase 4: 개발자 온보딩 및 구 저장소 아카이빙	- 전 개발자 대상 모노레포 명령어/워크플로우 온보딩 세션 진행 - 기존 8개 개별 Git 저장소 Read-only(Archived) 전환 완료	**4주차 (3 M/D)**	- **전사 모노레포 전환 완료 보고서** - CI 러너 비용 70% 이상 절감 지표 확인

6.2 참고 자료 (References)

- Turborepo Official Documentation: Structuring a Repository & Remote Caching
- pnpm Workspaces & Catalogs (`catalog:` protocol) Specification
- Monorepo.tools: In-depth Comparison of Turborepo, Nx, Bazel, and Lerna
- Dockerizing a Turborepo with `turbo prune` for Minimal Production Images