HYUNDAI AUTOEVER

GITLAB AGENTIC WORKFLOWS

내부망 AI 코딩 에이전트 플랫폼 구축 전략

GitLab CI/CD + LLM Agent Pipeline · Issue-to-MR 자동화 · 보안 샌드박스 실행
GitHub Agentic Workflows의 내부망 GitLab 구현 전략

65%
개발 생산성 향상

ROI 4.2x
투자 대비 수익

3.5h→40m
이슈 해결 시간

연 8.4억
비용 절감

01 — BENCHMARK

GitHub Agentic Workflows 분석

GitHub Copilot Coding Agent의 핵심 기능과 워크플로우

Coding Agent (GA)

Issue 할당 → 자동 코드 분석 → 구현 → 테스트 → PR 생성. GitHub Actions 샌드박스 비동기 실행. 2025.09 정식 출시.

Markdown Workflow

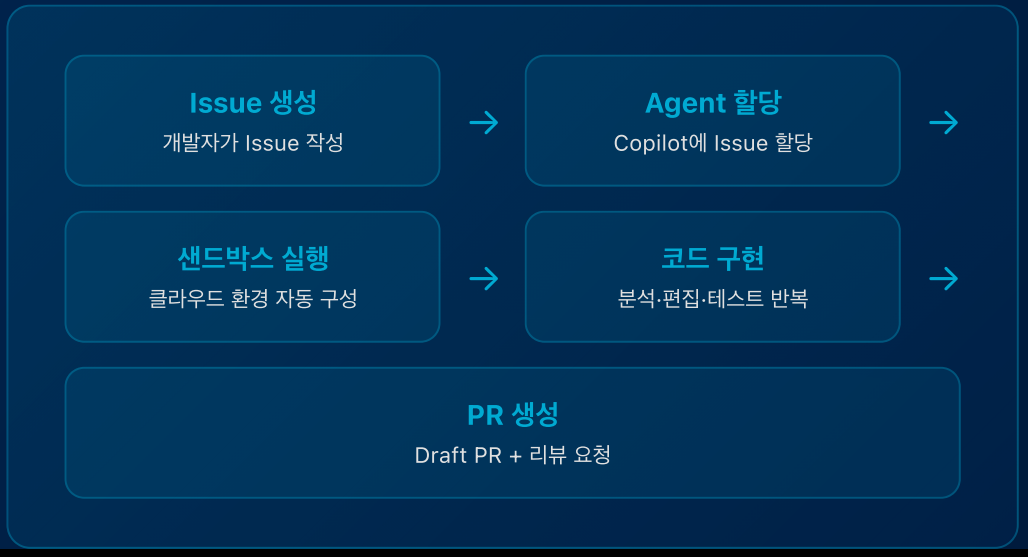
복잡한 YAML 대신 **Markdown 파일**로 워크플로우 정의. 자연어 지시를 GitHub Actions로 자동 변환. 직관적 설정.

Multi-Agent

Claude, Codex 등 서드파티 에이전트 통합. MCP 기반 확장. 엔터프라이즈 감사 로그 + 거버넌스 관리.

보안 샌드박스

기본 읽기 전용 권한. 쓰기는 사전 승인된 출력만 허용. 최소 권한 원칙 (Least Privilege) 적용.



02 — WHY GITLAB

왜 내부망 GitLab 에이전트인가

GitHub 대비 내부망 GitLab 환경의 필요성과 차별점

GitHub Copilot Agent의 한계

- 코드가 GitHub 클라우드 샌드박스로 전송
- 내부 소스코드 외부 노출 리스크
- 사내 인프라(SAP, DMS 등) 연동 불가
- 엔터프라이즈 보안 정책 충족 어려움
- 모델 선택 제한 (GitHub 제공 모델만)

내부망 GitLab Agent의 장점

- 폐쇄망 내 전체 파이프라인 실행
- 소스코드 외부 유출 Zero
- 사내 시스템 직접 연동 (SAP, Jira, DMS)
- SOC2, ISO27001 기반 보안 정책 준수
- 자체 LLM + Claude API 하이브리드 선택

100%
데이터 폐쇄망 처리

Zero
외부 코드 유출

직접
사내 인프라 연동

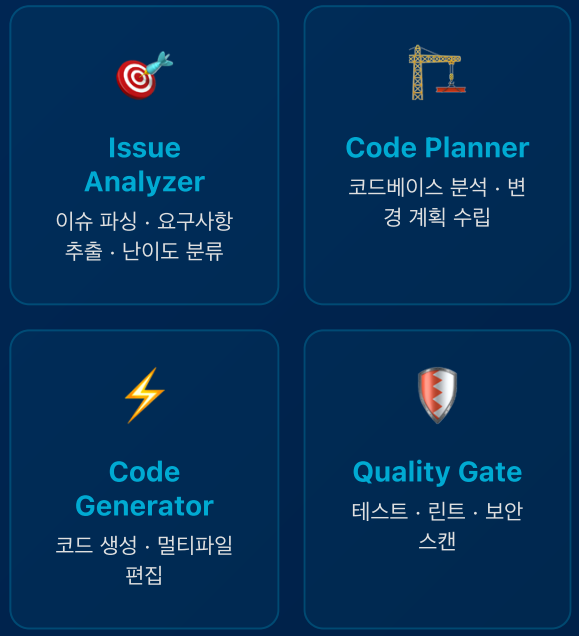
자유
모델 선택 유연성

03 — ARCHITECTURE

핵심 아키텍처

GitLab CI/CD + AI Agent 파이프라인 전체 구조

Agent Orchestration Layer



Infrastructure Stack

Layer	Technology
SCM	GitLab Self-Managed (CE/EE)
CI/CD	GitLab Runner + Docker Executor
Sandbox	격리 Docker 컨테이너 (per-task)
AI Engine	Claude API / vLLM (자체 호스팅)
Agent Framework	Claude Agent SDK / LangGraph
MCP Server	GitLab MCP + 사내 시스템 MCP
Storage	PostgreSQL + Redis + MinIO
Monitoring	Grafana + Prometheus + Sentry

04 — PIPELINE

Issue-to-MR 자동화 파이프라인

이슈 생성부터 Merge Request까지 완전 자동화 워크플로우

1

Issue Trigger Webhook

개발자가 GitLab Issue에 `/agent` 라벨 또는 `@ai-agent` 멘션 → GitLab Webhook이 Agent Orchestrator에 이벤트 전송

2

Context Analysis Sonnet

이슈 본문 + 코멘트 + 코드베이스 구조 분석 → 변경 필요 파일 식별, 영향 범위 산정, 구현 계획(Spec) 수립

3

Sandbox Execution Docker

격리된 Docker 컨테이너에서 리포지토리 클론 → 코드 생성/수정 → 빌드 → 단위 테스트 → 린트 실행. 실패 시 Self-healing 루프

4

Quality Gate 필수

CI 파이프라인 통과 · SAST/DAST 보안 스캔 · 코드 커버리지 80%+ · 아키텍처 규칙 검증 → 모두 Pass 시 MR 생성

5

Merge Request Draft MR

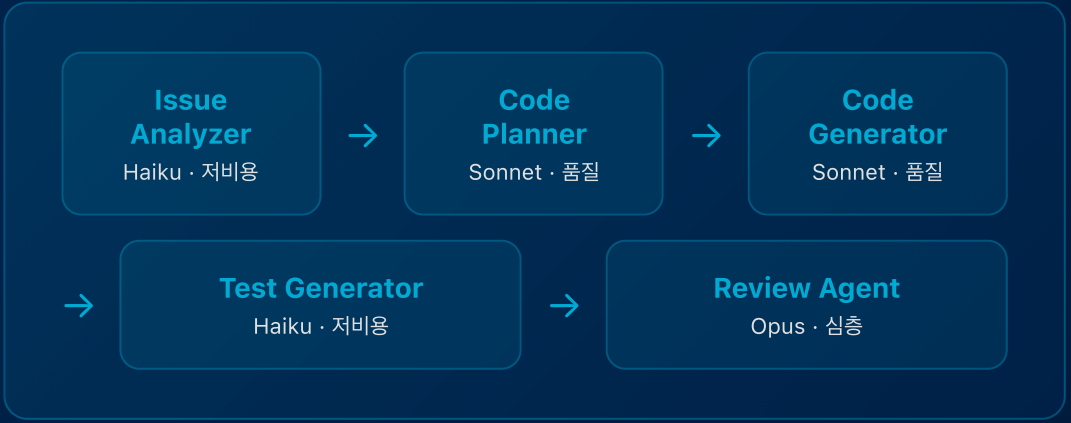
Draft MR 자동 생성 → 변경 요약 · 테스트 결과 · 영향 분석 포함 → 담당 리뷰어 자동 배정 → 리뷰 코멘트 반영 루프

05 — AGENTS

5-에이전트 파이프라인 구성

각 에이전트의 역할과 LLM 모델 배치 전략

Agent	Model	역할	입력	출력
Issue Analyzer	Haiku	이슈 파싱, 요구사항 추출, 난이도/복잡도 분류, 우선순위 판정	Issue JSON	Task Spec
Code Planner	Sonnet	코드베이스 분석, 변경 파일 식별, 구현 계획 수립, 의존성 그래프	Task Spec + Repo	Change Plan
Code Generator	Sonnet	코드 생성/수정, 멀티파일 편집, Self-healing (빌드 에러 자동 수정)	Change Plan	Code Diff
Test Generator	Haiku	단위/통합 테스트 자동 생성, 엣지 케이스 커버리지 확보	Code Diff	Test Suite
Review Agent	Opus(월간)	코드 품질 리뷰, 보안 취약점 분석, 아키텍처 컨벤션 검증	Code + Tests	MR + Review



06 — SECURITY

보안 샌드박스 아키텍처

격리된 실행 환경과 다층 보안 체계

컨테이너 격리

에이전트 태스크당 격리된 Docker 컨테이너 생성. 네트워크 정책으로 내부망만 접근 허용. 태스크 완료 후 자동 삭제. CPU/메모리 리소스 제한.

코드 실행 제한

읽기 전용 마운트 (원본 리포지토리). 쓰기는 임시 워크스페이스만 허용. 외부 네트워크 차단 (사내 레지스트리만 허용). 실행 시간 타임아웃 (30분).

감사 추적

모든 에이전트 액션 감사 로그. LLM 프롬프트/응답 전체 저장. Git 커밋 히스토리 추적. 관리자 대시보드에서 실시간 모니터링.

보안 레이어

네트워크 격리	완전
컨테이너 격리	완전
SAST/DAST 스캔	자동
시크릿 검출	완전
실행 타임아웃	30분
감사 로깅	완전
휴먼 리뷰 게이트	필수

토큰 최적화 & 알려진 한계

에이전트 비용 효율화와 AI 코딩 에이전트의 현실적 한계 대응

cgrep: 시맨틱 코드 검색

BM25(Tantivy) + AST 분석(tree-sitter) 하이브리드 코드 검색. 에이전트가 코드베이스를 탐색할 때 **95% 토큰 절감**, 58배 속도 향상. MCP Server로 에이전트에 직접 연동.

2단계 에이전트 워크플로우

1단계: cgrep으로 후보 파일/함수 로케이팅
2단계: 선택적으로 필요한 코드만 확장 로딩
불필요한 전체 파일 읽기를 제거하여 **토큰 비용 20배 절감**

프롬프트 캐싱

반복 패턴 프롬프트 캐시 · 프로젝트 컨텍스트 프리로딩 · 에이전트별 시스템 프롬프트 최적화. 월간

AI 코딩 에이전트의 알려진 한계

의존성 버전 환각	높음
단순 문자열 치환	중간
컨텍스트 윈도우 한계	중간
대규모 리팩토링 실패	높음

완화 전략: 버그 수정 · 테스트 생성 · 문서 업데이트 등 **범위가 명확한 작업**에 에이전트 집중. 대규모 리팩토링 · 아키텍처 변경은 **개발자가 직접 수행**. lock 파일 기반 의존성 검증 필수.

API 비용 40% 추가 절감.

07 — USE CASES

핵심 활용 시나리오

에이전트가 자율 처리하는 개발 업무 유형

버그 수정 자동화

Before: 3.5시간 → **After:** 40분

81% 절감

에러 로그 분석 → 원인 파일 식별 → 수정 코드 생성 → 회귀 테스트 → Draft MR

테스트 커버리지 확대

Before: 2일 → **After:** 2시간

87% 절감

커버리지 갭 분석 → 테스트 케이스 자동 생성 → 엣지 케이스 포함 → 80%+ 달성

문서 자동 업데이트

Before: 수동/누락 → **After:** 자동

누락 Zero

코드 변경 감지 → API 문서 · README · CHANGELOG 자동 갱신

기능 구현 보조

Before: 1주 → **After:** 1-2일 70% 절감

이슈 요구사항 → 구현 계획 → 스캐폴딩 코드 생성 → 테스트 포함 → 리뷰 대기

기술 부채 해소

Before: 후순위 → **After:** 자동 처리

지속 개선

코드 스멜 감지 → 리팩토링 제안 → 자동 수정 → 회귀 테스트 검증

보안 취약점 패치

Before: 4시간 → **After:** 30분

88% 절감

SAST 스캔 결과 → 취약점 분류 → 패치 코드 생성 → 보안 테스트 → MR

08 — ROI

ROI 분석

개발 생산성 향상과 비용 절감 효과

항목	Before	After	절감
버그 수정	3.2억원	0.6억원	2.6억원
테스트 작성	2.8억원	0.4억원	2.4억원
기술 부채 해소	1.5억원	0.3억원	1.2억원
문서 관리	0.8억원	0.1억원	0.7억원
보안 패치	1.2억원	0.2억원	1.0억원
합계	**9.5억원**	**1.6억원**	**7.9억원**

65%
개발 시간 절감

4.2x
투자 대비 ROI

8개월
투자 회수 기간

8.4억
연간 순 절감

정성적 효과

개발자 만족도 향상 (반복 업무 감소) · 코드 품질 일관성 확보 · 보안 취약점 사전 차단 · 온보딩 가속화 (신규 개발자 생산성 2배)

09 — COST

비용 구조

연간 운영 비용과 AI API 비용 상세

연간 운영 비용

항목	비용	비율
AI API (Claude)	0.9억원	47%
GitLab Runner 인프라	0.4억원	21%
운영 인력 (2명)	0.3억원	16%
모니터링/보안 도구	0.2억원	11%
교육/온보딩	0.1억원	5%
합계	**1.9억원**	**100%**

투자 **1.9억** → 절감 **8.4억** = **ROI 4.2x**

AI 비용 (1,000회 실행 기준)

Agent	Model	비용
Issue Analyzer	Haiku	$0.50
Code Planner	Sonnet	$4.50
Code Generator	Sonnet	$6.00
Test Generator	Haiku	$0.50
Review Agent	Opus	$45.00
합계		**$56.50**

비용 최적화 전략

Haiku 우선: 고빈도 분류/분석 작업
Sonnet 핵심: 코드 생성/계획 품질 확보
Opus 제한: 월간 심층 리뷰만 사용
캐싱: 반복 패턴 프롬프트 캐시 활용

10 — EXTENSION

MCP 연동 & 사내 시스템 확장

Model Context Protocol 기반 사내 인프라 통합

GitLab MCP Server

Issue/MR CRUD · 파이프라인 상태 조회 · 코드 리뷰 코멘트 · 브랜치 관리 · CI/CD 트리거. 에이전트가 GitLab을 직접 조작.

Confluence MCP

사내 위키 검색 · 기술 문서 참조 · 아키텍처 문서 조회 · 온보딩 가이드 참조. RAG 기반 지식 보강.

Jira MCP Server

프로젝트 이슈 연동 · 스프린트 상태 조회 · 이슈 자동 생성/업데이트 · 에픽-스토리 매핑.

SAP MCP Server

비즈니스 로직 참조 · 데이터 모델 조회 · RFC/BAPI 인터페이스 문서. 도메인 컨텍스트 제공.

SonarQube MCP

코드 품질 메트릭 · 보안 취약점 리포트 · 기술 부채 분석 · 커버리지 데이터 조회.

사내 DB MCP

테이블 스키마 조회 · 샘플 데이터 참조 · 쿼리 검증. 읽기 전용 접근으로 보안 유지.

MCP (Model Context Protocol) — AI 에이전트가 외부 도구와 데이터 소스에 표준화된 방식으로 접근하는 프로토콜. 사내 시스템마다 MCP Server를 구현하면 에이전트가 **코드 생성 시 실시간으로 사내 컨텍스트를 참조**하여 도메인 정확도를 극대화합니다.

11 — ROLLOUT

4단계 롤아웃 전략

파일럿에서 전사 플랫폼까지 단계적 확산

P1

파일럿 (1-3개월) MVP

1개 팀 (5명) · 버그 수정 + 테스트 생성 자동화
GitLab Runner + Claude API 연동
목표: 주 20건 이슈 자동 처리

P2

확장 (4-6개월) Scale

3개 팀 (15명) · 기능 구현 보조 추가
MCP 연동 (Jira, Confluence)
목표: 월 200건 이슈 자동 처리

P3

부서 확대 (7-12개월) Dept

개발 본부 전체 (50명+) · 보안 패치 자동화 추가
SAP/DB MCP 연동 · 커스텀 에이전트 개발
목표: 월 500건 자동 처리

P4

전사 플랫폼 (12개월+) Enterprise

그룹사 전체 개발 조직 · 셀프 서비스 에이전트 마켓
커스텀 에이전트 빌더 · 멀티 모델 지원
목표: 전사 AI 개발 플랫폼

5명	15명	50+명	전사
P1: 파일럿 팀	P2: 확장	P3: 본부	P4: 플랫폼

12 — RISK

리스크 관리

예상 리스크와 완화 전략

리스크	확률	영향	완화 전략
AI 생성 코드 결함	높음	높음	필수 CI 파이프라인 (테스트 + SAST) + 휴먼 코드 리뷰 게이트 + 프로덕션 직접 머지 금지
보안 취약점 주입	중간	높음	SAST/DAST 필수 통과 + 시크릿 스캔 + 네트워크 격리 샌드박스 + 보안팀 리뷰 게이트
API 비용 초과	중간	중간	Haiku 우선 전략 + 월간 비용 상한 + 프롬프트 캐싱 + 사용량 모니터링 대시보드
개발자 저항	중간	중간	파일럿 성공 사례 공유 + 점진적 도입 + AI는 보조 도구 포지셔닝 + 챔피언 개발자 육성
모델 의존성	낮음	중간	멀티 모델 아키텍처 (Claude + vLLM 폴백) + model_version 태깅 + 프롬프트 GitOps 관리

핵심 원칙: AI 생성 코드는 반드시 CI 파이프라인 + 휴먼 리뷰를 통과해야 머지. **프로덕션 직접 배포 금지**. 모든 에이전트 행동은 감사 로그로 추적. 에이전트는 보조 도구, **최종 판단은 개발자**가 합니다.

13 — ROADMAP

90일 실행 로드맵

인프라 구축부터 정식 운영까지

주차	목표	산출물	성공 기준
1-2	인프라 구축 + 보안 검증	GitLab Runner 구성, 샌드박스 Docker, 보안 검증	보안팀 감사 통과
3-4	Agent Pipeline v0.1	Issue Analyzer + Code Generator 작동, CI 연동	버그 수정 자동화 정확도 >80%
5-6	파일럿 시작	1개 팀 온보딩, 버그 수정 + 테스트 생성 자동화	주 10건 자동 처리
7-8	파일럿 확대	Quality Gate 강화, MCP 연동 (Jira, Confluence)	주 20건 자동 처리
9-10	품질 최적화	피드백 반영, 프롬프트 튜닝, 에이전트 정확도 향상	개발자 만족도 >4.0
11-12	정식 운영 전환	운영 매뉴얼, KPI 대시보드, 확장 계획	월 80건 자동 처리

4주
MVP 완성

8주
파일럿 안정화

12주
정식 운영

HYUNDAI AUTOEVER

GITLAB AGENTIC WORKFLOWS

개발자의 시간을 창의적 문제 해결에 집중시킵니다

반복 코딩은 AI 에이전트가 처리하고,
개발자는 설계, 아키텍처, 비즈니스 로직에 집중합니다.

4주	8주	12주
MVP 완성	파일럿 시작	정식 운영

AI 에이전트는 보조 도구, **최종 판단은 개발자**가 합니다.