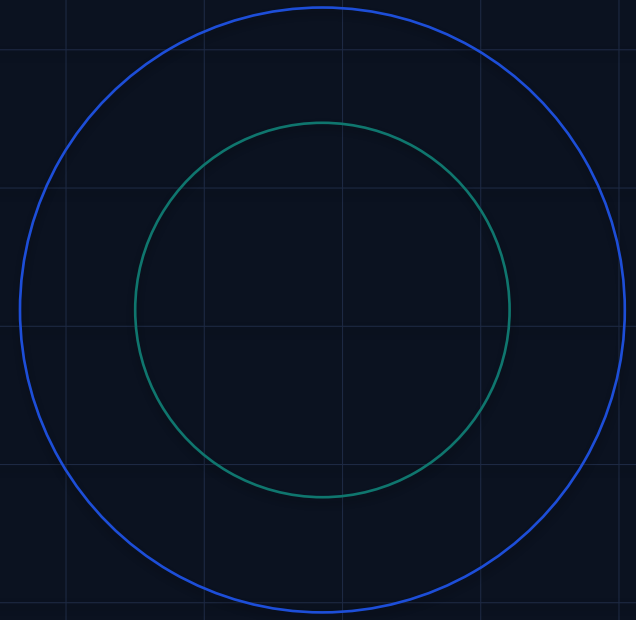
PRODUCT OVERVIEW · 2026

VersionMaster

폐쇄망을 위한 올인원 Git Forge

형상관리 · 코드리뷰 · CI/CD · 프로젝트 관리
망분리 반입·반출 통제 · Local AI 코드리뷰



주식회사 피피티웨어 | www.pptware.com

On-Premise · Korean-first · Local AI

WHY VERSIONMASTER

개발은 빠르게, 통제는 확실하게

폐쇄망에서 필요한 기능을 하나의 제품으로

01

통제

· 반입·반출 승인 워크플로
· 서명 번들·무결성 검증
· 감사 해시 체인

요청 → 승인 → 반출 → 감사

02

생산성

· MR·인라인 코드리뷰
· CI/CD·릴리스 게이트
· 프로젝트·WBS·간트
· 사내 패키지 레지스트리

Plan → Code → Review → Release

03

운영

· Docker/Helm 단일 스택
· 폐쇄망 오프라인 설치
· 진단·백업·HA

설치 → 진단 → 복구 → 관측

GitHub/GitLab/Gitea + 별도 반출·프로젝트·AI 도구를 하나로 대체

CLOSED-NETWORK REALITY

폐쇄망 개발은 '저장소'만으로 해결되지 않습니다

현장에서 반복되는 네 가지 병목

01

반출 통제가 저장소 밖에 있다

승인·반출·감사를 별도 DLP/수작업으로 처리

02

도구가 분산된다

Git + Jira + CI + Registry + 반출 도구를 각각 운영

03

한글 레거시가 깨진다

CP949/EUC-KR 파일의 열람·diff·검색 실패

04

외부 AI를 쓸 수 없다

코드 반출 금지로 클라우드 AI 리뷰 불가

VersionMaster는 이 네 가지를 한 제품 안에서 해결합니다.

ONE PLATFORM

VersionMaster 한눈에 보기

Git을 중심으로 개발·협업·통제·운영을 통합

VersionMaster

폐쇄망 전용 올인원 Git Forge

외부 SaaS 없이 사내에서
설치·운영·감사까지 완결

DEPLOY Docker Compose / Helm

MODE On-Premise / Offline

CORE Go API + Rust libgit2

Git & MR

저장소·브랜치·MR·인라인 리뷰

CI/CD

파이프라인·잡·릴리스·게이트

Projects

보드·WBS·간트·이슈

Transfer Gate

승인·서명 번들·감사

Registry & LFS

npm·Maven·Docker·대용량

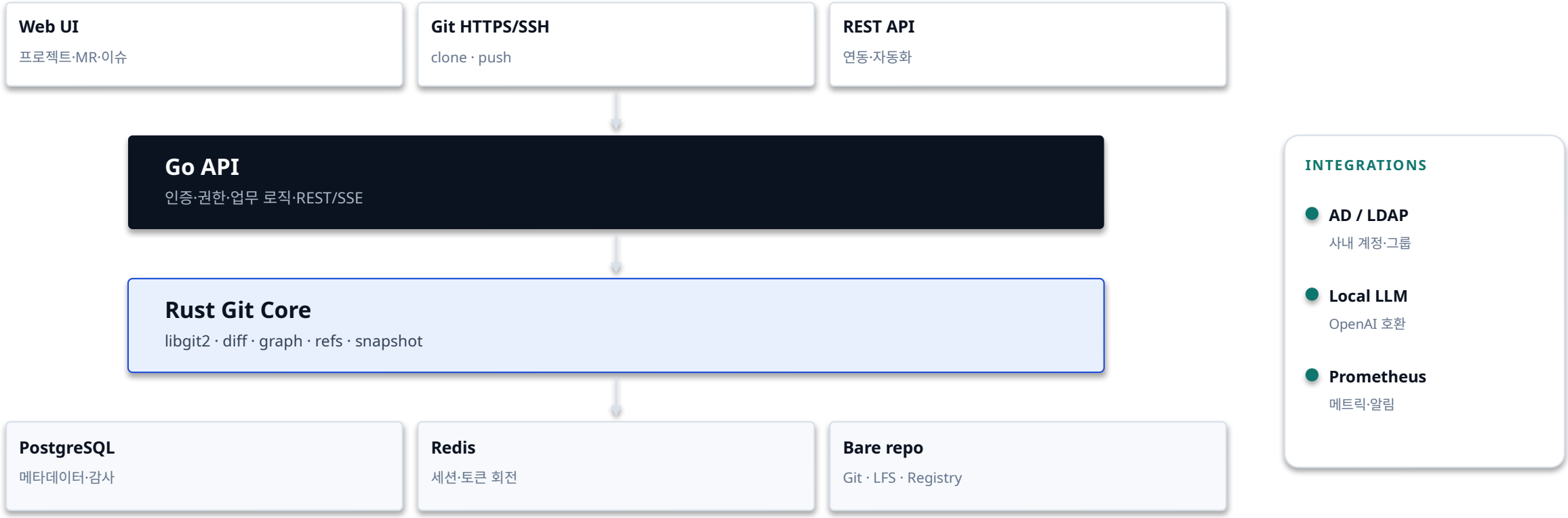
Local LLM

사내 모델 코드리뷰·제안 적용

ARCHITECTURE

검증된 단일 스택, 폐쇄망에 맞춘 구조

Go API + Rust Git Core + PostgreSQL/Redis



외부 SaaS 의존 없이 Docker Compose / Helm / 오프라인 번들로 설치

DIFFERENTIATOR 01

망분리 반입·반출을 제품 안에서 통제

요청부터 완료까지 승인 워크플로 + 서명 번들 + 감사

1 **요청**
목적·파일·대상

2 **승인**
SoD·정책 검사

3 **패키징**
서명 번들

4 **반출**
무결성 검증

5 **반입**
대상 검증

6 **완료**
감사 기록

SoD
요청자 ≠ 승인자

Secret Scan
개인키·.env·자격증명 차단

Signed Bundle
HMAC + SHA-256

Audit Chain
해시 체인으로 위변조 탐지

반출 번들 구성

승인 시점의 파일 목록과 서명을 함께 묶어 위변조를 차단

manifest.json
파일 목록·메타데이터

checksums.sha256
파일별 무결성 해시

signature.hmac
승인자 서명

approval.json
승인 이력·정책 결과

요청·승인·해시·서명이 감사 로그에 함께 기록되어 반출 이력을 재구성할 수 있습니다.

반출 통제를 위해 별도 DLP 제품을 붙일 필요가 없습니다.

GOVERNANCE MAP

사용자와 운영자의 통제 접점

가입부터 복구까지, 각 역할이 확인하는 증적

사용자 여정

01 **가입·승인**
본명·조직·프로젝트 신청 → 관리자 승인

02 **권한 확인**
조직/팀/직접 멤버 중 최대 권한

03 **변경 리뷰**
MR·지정 승인자·CI/AI 게이트

04 **반출 요청**
사유·ref·산출물 선택 후 승인

05 **감사 확인**
요청 이력·해시 체인 무결성

운영자 여정

01 **설치·키**
오프라인 번들·JWT/서명키·vm-doctor

02 **계정·조직**
가입 승인·조직/팀·저장소 위임

03 **정책·스캔**
금지 경로/용량·시크릿 스캔

04 **관측·SLA**
승인 적체·감사 체인 alert

05 **백업·복구**
4종 자산·SHA·복원 리포트

같은 사건을 사용자는 '요청'으로, 운영자는 '증적'으로 본다.

USE CASE 01

반출 요청과 승인, 분리 권한이 핵심

요청자와 승인자가 다르지 않으면 승인은 효력이 없다

1 **요청**
사유·ref·산출물

2 **제출**
승인 대기

3 **검토**
정책·시크릿 스캔

4 **승인/반려**
SoD 강제

5 **패키징**
서명 번들

SoD

요청자 ≠ 승인자
자기 승인은 403

저장소 관리자

owner/admin/팀 admin만
승인·반려 가능

감사 증적

요청·승인·패키징
모두 해시 체인 기록

승인 화면에서 함께 보는 증적

- 요청 사유·ref
- 정책·시크릿 스캔
- 감사 이력·무결성
- 승인 N/M · SoD

요청자 본인 승인은 403 SEPARATION_OF_DUTIES — 승인 절차를 형식이 아니라 통제로 만든다.

USE CASE 02

번들 생성부터 반입까지, 4단계 검증

서명·해시·저장소 대조를 모두 통과해야 적용된다

반출 번들 (.vmx) 구성

manifest.json	manifest.sig	patches/	files/	git.bundle
메타·해시	HMAC 서명	커밋별 패치	스냅샷	오브젝트+refs

1 구조 검증
필수 파일 존재

2 해시 검증
SHA-256 대조

3 서명 검증
HMAC-SHA256

4 git bundle
verify / fetch

1순위 · git bundle
빈 저장소도 원본과 동일한 SHA 이력

2순위 · snapshot
대상 저장소가 비었을 때 파일로 부트스트랩

3순위 · patch
증분 적용, 충돌 시 원본 저장소는 무변경

승인된 것만, 원본과 동일한 이력으로 반입 — 서명·해시·저장소 대조를 모두 통과해야 한다.

USE CASE 03

운영자는 '증적'으로 거버넌스를 완성한다

감사·모니터링·복구가 한 흐름으로 이어져야 한다

감사 · 모니터링

- **해시 체인** — 모든 반출입 단계 prev_hash/hash
- **외부 앵커** — WORM/NAS에 JSONL append-only
- **주기 검증** — 30초마다 체인 검증 · broken alert
- **승인 SLA** — 24시간/3일 적체 alert · webhook
- **SIEM · TSA** — 감사 export · 시점 증명

백업 · 복구

01 **백업**
DB + repos + registry + LFS + artifacts

02 **검증**
SHA256SUMS · MISSING-COMPONENTS

03 **복원**
4종 자산을 DATA_DIR 기준으로

04 **리포트**
git rev-parse · vm-doctor · report

감사 로그는 DB 밖 앵커·SIEM, 복구는 4종 자산과 리포트까지 — 운영의 증적이 남아야 한다.

POC CHECKLIST

PoC에서 이 6가지를 확인하세요

기능 시연이 아니라 '증적'을 남기는 PoC

01 가입 승인

pending 계정은 로그인·API 모두 차단

→ approval_status

02 분리 권한

요청자 자기 승인은 403

→ SEPARATION_OF_DUTIES

03 시크릿 차단

개인키 포함 번들은 생성 거부

→ SECRET_DETECTED

04 번들 변조

서명·해시 불일치 번들은 반입 거부

→ VERIFICATION_FAILED

05 감사 무결성

audit_integrity=true · broken alert

→ audit_integrity

06 복원 리포트

4종 자산 복원 + SHA/보고서

→ restore-report

PoC 산출물: 승인 이력 · 번들 SHA-256 · 감사 체인 · 복원 리포트

DIFFERENTIATOR 02

한국 기업의 개발 현실에 맞춘 기본기

계정·조직·레거시 코드까지 한국형 운영 규칙을 기본 제공

01

CP949 / EUC-KR

레거시 한글 파일의 열람·diff·검색을 깨지지 않게 처리

02

AD / LDAP

사내 계정으로 로그인하고 그룹을 팀 권한에 자동 매핑

03

조직 계층

대표조직 > 사업부 > 그룹 > 팀 > 파트 > 팀원

04

가입 승인

관리자 승인 후 로그인 가능, AD 자동 생성 예외

글로벌 포지가 놓치는 국내 운영 규칙을 기본 제공합니다.

DIFFERENTIATOR 03

Git Forge 안에서 끝나는 프로젝트 관리

여러 저장소를 하나의 업무 단위로 묶고 계획·실행·진척을 한 화면에

01

프로젝트

여러 저장소를 업무 단위로 묶고
조직·담당·우선순위를 연결

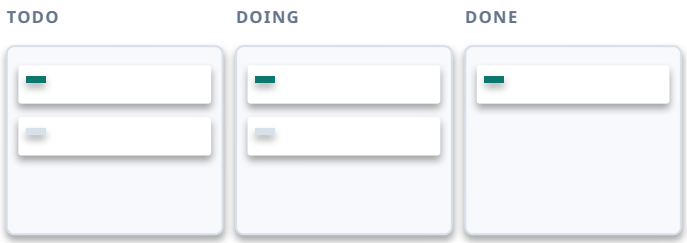


여러 저장소를 하나의 업무 단위로

02

보드 · WBS

칸반·계층·의존성·마감일·진행률을
이슈와 저장소에 직결



03

간트

일정 막대 드래그·리사이즈·진행률,
오늘선으로 지연을 즉시 확인



오늘선으로 지연을 즉시 확인

Jira·MS Project 없이 프로젝트 현황을 한눈에 확인합니다.

DIFFERENTIATOR 04

코드가 외부로 나가지 않는 AI 코드리뷰

사내 Local LLM으로 MR을 자동 리뷰하고, 사람이 최종 판단

1 MR 생성 · push
자동 트리거

2 Diff 필터 · 청킹
락·바이너리 제외

3 Local LLM
OpenAI 호환

4 인라인 코멘트
파일·줄 단위

제안 적용
한 줄 코드 제안을 버튼으로 커밋

후속 질문
스레드 맥락으로 AI가 재답변

피드백
유용/불필요 평가로 수집

병합 게이트
리뷰 완료 전 병합 차단(선택)

MR !128 · AuthService.java

```
41 - if (token.isExpired()) return null;
42 + if (token.isExpired()) { audit.log("expired"); return null; }
```

만료 토큰의 감사 로그가 누락되어 있습니다.
회전 이력과 함께 검증하도록 수정합니다.

AI REVIEW · LOCAL LLM

토큰 만료 검증은 유지하면서
감사 추적을 추가한 패치를 제안합니다.

제안 적용 사람이 검토 후 커밋

Guardrails · 시크릿 마스킹 · 프롬프트 인젝션 방어 · AI 승인/병합 권한 없음

DIFFERENTIATOR 05

폐쇄망 빌드의 마지막 퍼즐, 공급망

인터넷 없이 의존성 해결부터 이미지·배포까지

01

npm · Maven · Docker

사내 레지스트리에서 패키지를 보관·배포하고 외부 의존성을 통제

02

CI 이미지 게이트

허용된 사내 이미지만 실행, 외부 이미지 pull 차단

03

Git LFS

설계도·PLM·빌드 산출물 등 대용량 파일을 같은 저장소·권한으로

04

오프라인 설치

이미지·의존성·설치 번들을 반입해 폐쇄망에서 설치

인터넷 없이 의존성 해결부터 배포까지 이어집니다.

DIFFERENTIATOR 06

감사 가능한 개발 플랫폼

인증·권한·시크릿·감사를 기능으로 내장

01

Hybrid JWT

access 메모리 + HttpOnly refresh 쿠키 / Redis 회전

02

PAT Scope

read · write · admin 범위를 토큰 단위로 제한

03

2FA · 승인

TOTP, 가입 승인, 로그인 시도 제한

04

Secret Encryption

저장소 시크릿 AES-256-GCM 암호화

05

Audit Chain

감사 로그 해시 체인으로 위변조 탐지

06

Guardrails

SSRF 차단, 용량·파일 크기·AI 리뷰 안전장치

DIFFERENTIATOR 07

운영 부담을 줄이는 배포·관리

설치·진단·복구·관측을 한 제품에

01

설치

Docker Compose, Kubernetes/Helm,
폐쇄망 오프라인 번들

02

진단

vm-doctor로 설정·키·권한·인증서·연결 점검

03

복구

백업·복원, HA/페일오버,
블루/그린 배포 구성

04

관측

Prometheus 메트릭, Grafana 대시보드, 알림 규칙

단일 스택으로 설치·운영·복구의 복잡도를 낮춥니다.

POSITIONING

어떤 Git Forge와 무엇이 다른가

공개 기능 기준 비교 · 구성에 따라 달라질 수 있음

	VersionMaster	GitHub	GitLab	Gitea/Forgejo	Azure DevOps
폐쇄망·오프라인 설치	●	◐	●	●	◐
반입·반출 승인 게이트	●	○	○	○	○
CP949/EUC-KR 처리	●	○	○	◐	○
AD/LDAP + 조직 계층	●	●	●	◐	●
프로젝트/WBS/간트	●	◐	●	◐	●
Local LLM 코드리뷰	●	○	◐	○	○
사내 패키지 레지스트리	●	◐	●	◐	●
감사 해시 체인	●	◐	◐	◐	◐
운영 복잡도	낮음	중간	높음	낮음	중간

● 기본 제공 ◐ 부분 지원·별도 구성 ○ 미제공 또는 외부 제품

2026년 공개 기능 기준. GitHub Enterprise / GitLab Self-Managed / Azure DevOps Server 등 설치형·유료 애드온에 따라 달라질 수 있습니다.

USE CASES

이런 조직에 특히 적합합니다

폐쇄망·규제·레거시·다중 저장소가 겹치는 환경

공공 · 국방

반출 통제·감사·완전 폐쇄망

승인·서명 번들·감사 체인으로 반출을 통제하고, 외부 SaaS 없이 운영

금융

AD/2FA·감사·내부 AI

사내 계정·2FA·감사 로그를 기본으로, 코드 외부 유출 없이 AI 리뷰

제조

대용량·레거시·협력사

LFS로 설계·산출물을 관리하고 CP949/EUC-KR 레거시를 그대로 수용

SI · 대기업

다중 저장소·프로젝트

프로젝트·WBS·간트로 여러 저장소와 협력사 작업을 한 화면에서

공통 기반: 폐쇄망 설치 · 감사 추적 · 권한 통제 · 단일 운영

OUTCOMES

도입 전후

도구 통합과 통제 일원화로 운영 방식을 바꿉니다

영역	As-Is	To-Be
반출	별도 DLP·수작업 승인	승인·서명·감사 내장
개발	Git + Jira + CI + Registry	단일 플랫폼
리뷰	외부 AI 사용 불가	Local LLM 자동 리뷰
운영	다수 구성요소·수동 점검	단일 스택·진단·백업

One platform · One audit trail · One deployment

HOW TO START

5단계 도입 프로세스

현황 진단부터 운영 이관까지 PPTware가 함께합니다

01

진단

현황·보안·조직 요구 분석

산출물

요구사항 정의서
현황 진단 보고서

02

PoC

핵심 시나리오 검증

산출물

PoC 시나리오
검증 결과

03

구축

설치·AD/CI/반출 연동

산출물

구축 완료 보고서
연동 체크리스트

04

이관

저장소·이슈·권한 이관

산출물

이관 계획
권한 매핑표

05

운영

교육·모니터링·백업

산출물

운영 가이드
모니터링·백업 정책

조직 규모·보안 요구에 따라 일정과 범위를 조정합니다.

LICENSE & SUPPORT

사내 독점 라이선스, 납품형 지원

설치·업그레이드·장애·보안까지 함께

01

License

· 사내 독점 라이선스
· 오픈소스 고지·라이선스 전문
· 폐쇄망 설치 문서

납품 시 라이선스 증서 제공

02

Support

· 설치·업그레이드 지원
· 장애 대응·원인 분석
· 보안 점검·운영 가이드

정기 점검·업그레이드 계획

03

Contact

· 도입 상담·PoC 지원
· 주식회사 피피티웨어
· www.pptware.com

제품 문의·기술 지원

VersionMaster · 폐쇄망 개발의 표준



VersionMaster

폐쇄망 개발의 표준

형상관리 · 코드리뷰 · CI/CD · 프로젝트 관리 · 반출 통제 · Local AI

회사

주식회사 피피티웨어

www.pptware.com

담당자

강대형 기술이사

john.kang@pptware.com

010-9237-1766

© 2026 PPTware Co., Ltd. All rights reserved.