TGO 鲲鹏會

KANBAN2 AGENT HARNESS

日产万行业务代码的Harness 之路

两个月，把 Kanban2 高吞吐交付托上跑道

分享人 | 吴穹博士 | 2026

LOOP 全球 AI 生态与创新峰会

LOOP 全球 AI 生态与创新峰会

FIELD EXPERIENCE

组织研发效能专家

工具会变，但组织难题很稳定：目标如何拆解、架构如何守住、团队如何协作、质量如何不掉线。今天的 Agent Harness，也是这个脉络里的新一层工程底盘。

01	**大型组织转型现场**	平安、招行、建信、华为、顺丰、蔚来等大型组织。
02	**产品技术管理**	工程体系、研发平台、团队协同，都放在一张图里看。
03	**AI Coding 落地观察**	个人变快以后，组织系统也必须一起升级。
04	**这次复盘视角**	用 Kanban2 实战，讲一条可执行的 Harness 建设路。

当 AI 让个人更快，真正要补的是组织的交付底盘。

TRANSFORMATION PROFILE

吴穹 Adam
北大软件工程博士 | Agilean 首席顾问

大型组织转型 | 产品技术管理 | 研发平台 | AI Coding | Agent Harness | 敏捷 / 精益

LOOP
全球 AI 生态与创新峰会

OPENING PROBLEM

个人提效了，组织交付却没跟上

很多团队已经看到 AI 让单个工程师写得更快：代码产出上去了，commit 变密了。但如果计划、评审、测试、运行时和部署没有同步升级，组织效能只会小幅改善，质量压力却会明显放大。

01	个人产出上去了	模型让写代码、改代码、补测试的局部速度明显变快
02	组织系统没同步	需求拆解、评审、验收、环境和发布仍按旧节奏运行
03	质量压力被放大	更多 diff、更密集变更、更难追溯，返工和线上风险变高

THE TRAP

个人速度不等于组织吞吐

没有 Harness，提效会先变成更多代码、更长等待和更多返工。

LOOP 全球 AI 生态与创新峰会

COMMON ANTI-PATTERNS

不要在底盘不稳时，急着追 Loop 和 Graph

Loop Engineering、Graph Engineering 是先进厂商实时 Agent 的下一步；但对大多数企业，第一步不是把智能体做得更会转圈，而是先把 Harness 做深，让反馈、验收、环境和权限稳定下来。

01	Harness 只停在脚本层	没有路径感知、真实运行时、失败证物和强制闸门
02	Agent feedback 太薄	模型不知道错在哪里，只能靠人肉截图和口头补充
03	人还没稳定提效，就上自治	人都跑不顺的流程，交给 Agent 只会更快地产生返工
04	用复杂编排掩盖地基问题	Loop/Graph 放大吞吐，也会放大不确定性和质量债

WRONG ORDER

先造自动智能体，再补 Harness

顺序反了，Agent 越能跑，组织越难知道它为什么错、错在哪里、该不该合。

正确顺序：先把人和 Agent 都能用的 Harness 做稳，再谈 Loop / Graph / 自治。

ROADMAP

目录

01	建设成效	Commits, churn, and pressure
02	建设之路	Agent OS, fullcheck, CLI E2E, skills, hooks
03	使用心得	When platform work starts compounding
04	落地建议	Reusable guardrails for high throughput

LOOP 全球 AI 生态与创新峰会

BUILD THE HARNESS ROAD

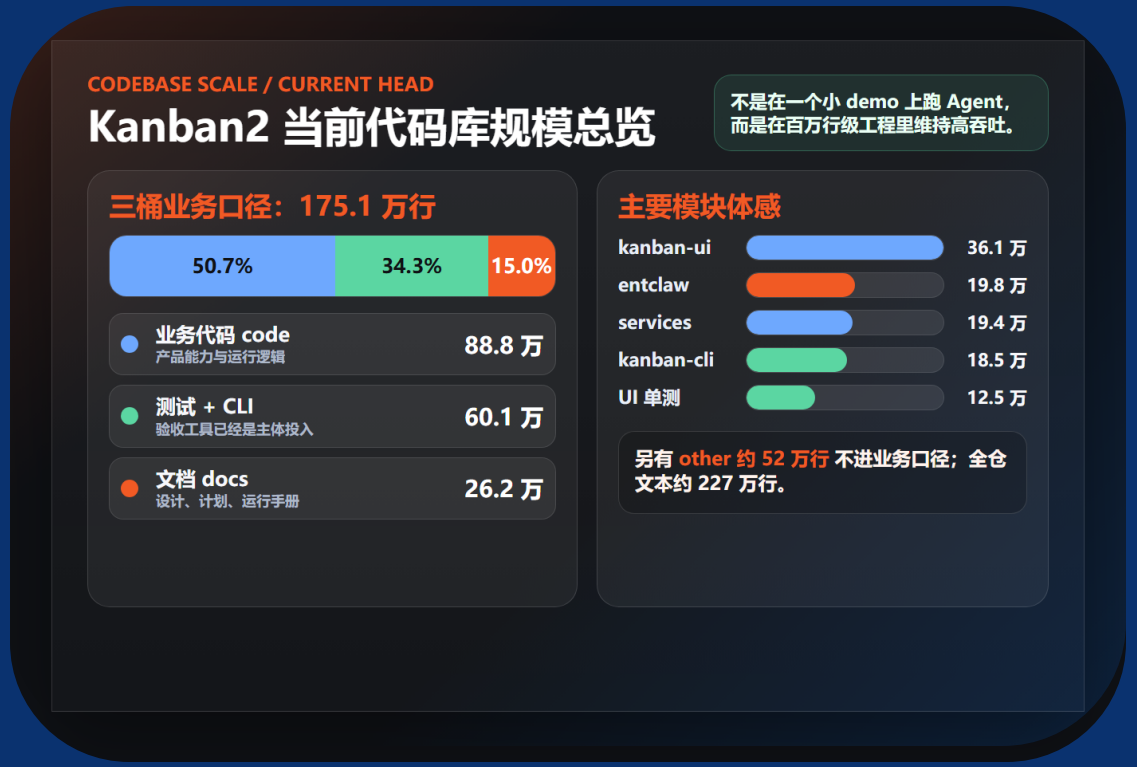
Kanban2 是百万行级工程

这不是小型演示项目上的阶段性实验；当前三桶业务口径已达 175.1 万行。

01 **175.1 万行**
三桶业务口径合计

02 **60.1 万行**
测试+CLI，占 34.3%

03 **18.5 万行 CLI**
验收工具已成子系统



01 / WORKDAY SNAPSHOT

先看一个普通工作日

一天切面：
多条任务线并行，
业务净增约 1.28 万行。

01	**52 次提交**	07:38-21:14
02	**+12,819 行**	业务代码净增
03	**7 条任务线**	同日推进

01 / COMMIT INTENSITY

高强度提交已经成为常态

前一页是一日切片；放回两个月看，高强度提交已经从峰值变成稳定巡航。

01	**1,818 次提交**	两个月持续推进
02	**1,007 核心业务**	触达核心路径
03	**35.4 次/日**	7 月活跃日均

01 / TWO-MONTH GROWTH

两个月稳定增长约 100 万行

主线不是一周增量，而是两个月多轮爬升：系统从几十万行推到 175 万行级。

01	**约 +100 万行**	两个月稳定增长
02	**175.3 万行**	2026-07-22 HEAD
03	**34.4% 测试+CLI**	护栏同步长厚

ROADMAP

目录

01	日产万行不是梦	Commits, churn, and pressure
02	Harness 建设之路	Agent OS, fullcheck, CLI E2E, skills, hooks
03	使用心得	When platform work starts compounding
04	落地建议	Reusable guardrails for high throughput

LOOP
全球 AI 生态与创新峰会

BUILD THE
HARNESS ROAD

9 步路线图：先修 Harness，再跑 Agent

每一步都把一种人肉流程，沉淀成 Agent 可执行、可观察、可复用的系统能力。

01 CLI 可编程验收

替代：验收只能靠手点，脚本散落，Agent 不会复跑。

02 数据 可复现宇宙

替代：手工造数、样本漂移，E2E 每次都在猜上下文。

03 测试 分层闸门

替代：质量回退靠人眼发现，失败只剩一句红灯。

04 观测 Agent 可读环境

替代：拷贝日志、截图喂模型，traceId 靠人翻。

05 环境 运行时自治

替代：手工启停、端口猜测，多 Agent 互相踩环境。

06 验收 UI 现场闭环

替代：API 绿了，界面还要忍受式手工验收。

07 上下文 文档文件系统

替代：上下文搬运靠聊天，换 Agent 就失忆。

08 技能 SOP 预装

替代：流程反复发明，经验留在人脑和聊天记录里。

09 优化 持续复利

替代：Harness 停在脚手架，flake 和慢反馈一直耗人。

核心顺序：先让 Agent 可验、可看、可起环境、可拿上下文，再谈多智能体自治。

01 / CLI HARNESS

1、构建完整CLI驱动验收

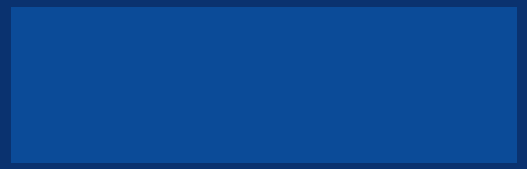
它有完整命令面，但主用户是 Agent、脚本和 fullcheck；命令行只是最薄的一层皮。

01 **127 / 143 有命令**
命令面真实存在，kanban 只是入口。

02 **17.0 万行 CLI**
主力沉在 setup / verify / 断言闭环。

03 **CLI-only 只有 3.9%**
多数是业务与验收同一次进仓。

没有它：验收只能靠手点，脚本散落，Agent 不会复跑。



命令行是皮，Harness 才是肉

02 / DATA FOUNDATION

2、利用CLI构造稳定测试数据

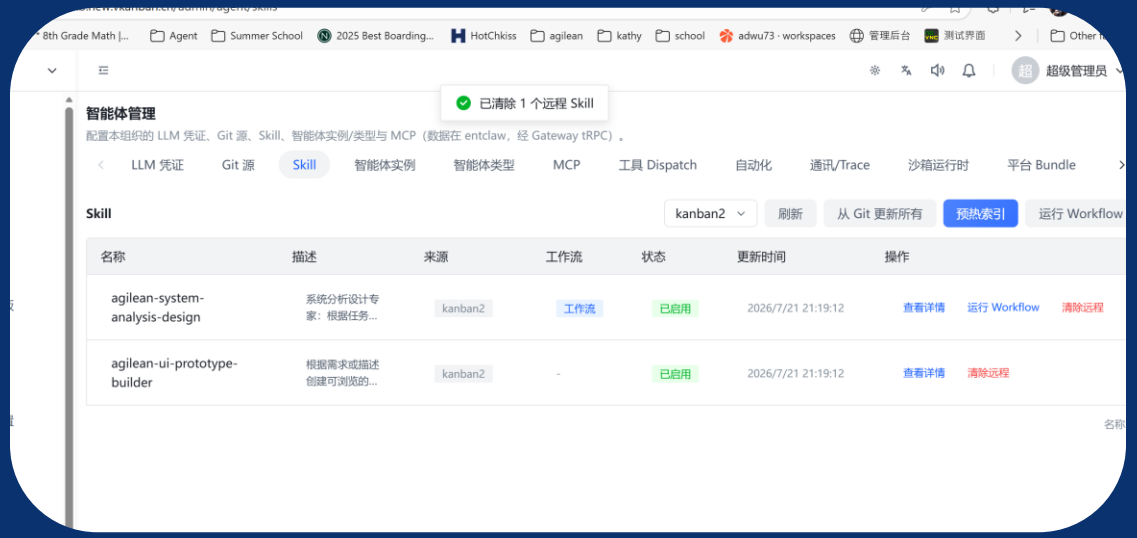
把稳定上下文、可复现样本和可预热能力放到底座里，Agent 不再临场拼数据。

01 数据可复现
固定账号、样本、组织和业务状态。

02 能力可预热
Skill 索引、Workflow 和运行时提前就绪。

03 断言可复用
同一套数据喂给 CLI、fullcheck 和 Agent。

没有它：手工造数、样本漂移，E2E 每次都在猜上下文。



Skill / Workflow / 预热索引，把测试数据接成可调度资产

3、构建分层 fullcheck 闸门

把 CLI E2E 收进 L0-L9 统一漏斗，路径感知决定跑哪几层，最后三关才碰真 Gateway。

01 **L0-L2 离线守卫**
runtime-sync / invariants / frontend。

02 **L3-L6 组件验证**
backend unit/IT、zgraph、CLI unit。

03 **L7-L9 真运行时**
Gateway API、CLI E2E、retry、L9 巡检。

没有它：fullcheck 只是命令集合，质量回退只能事后补锅。

02 / FULLCHECK EXAMPLE

fullcheck 实例

真实失败报告不是只给红绿灯：它告诉 Agent 哪层过了、哪层失败、失败点在哪里，以及下一步怎么复跑。

01 约 9 分钟跑完
从 L0 到 L8 输出分层摘要。

02 L0-L7 全绿
静态、前端、后端、API 先收口。

03 L8 精确失败
setup 96/138 定位到工时字段。

没有它：失败报告不可读，人要翻日志和数据库定位。

02 / COVERAGE BASELINE

fullcheck 跑出来的覆盖率水位

unit + IT + e2e-runtime 合并后，自有后端代码指令覆盖约 75.9%；这说明闸门已经形成基本质量水位。

01 **自有代码 75.9%**
排除第三方 Starlark 后；行覆盖 75.3%，分支 58.0%

02 **核心服务 73–79%**
card / meta / view 已经进入可用水位

03 **platform 是短板**
58.9% 指令覆盖，下一轮优先补测试与场景

没有它：只知道红绿，不知道覆盖短板和下一轮补哪里。

来源：build/jacoco/jacoco.xml，unit + IT + e2e-runtime 合并

BACKEND COVERAGE / JaCoCo

后端覆盖率已经有基本盘，下一步补平台短板

75.9%
自有代码指令覆盖
排除第三方 Starlark 后；行覆盖 75.3%

L7/L8
运行时验收也入账
unit + IT + e2e-runtime 合并口径

58.9%
platform 是短板
体量不小，下一轮优先补强

五大服务指令覆盖率

fullcheck 不是只跑“能不能编译”，它把单测、IT 和真运行时链路合成一张质量水位图。

Kernel / API / Infra

模块	指令	行	分支
domain	81.2%	80.8%	62.6%
event	87.3%	86.2%	64.7%
infra	78.7%	76.8%	65.5%
card-api	82.4%	83.2%	69.0%
schema-api	85.5%	83.6%	75.6%

全量含 vendored Starlark 为 69.0%；看自有代码时约 75.9%，更接近真实工程水位。

3.1 L1 invariants 是内核态权限检查

不等测试失败，先把错架构、文档漂移和危险写法挡在静态阶段。

01 工程卫生
生成文档、产物、大文件、旧端口先检查。

02 后端红线
Serializable、Yield.all、Action 命名等机器拦截。

03 前端红线
enum、非空断言、i18n、types 依赖先挡住。

没有它：能编译的错架构也能进仓，Review 变成人肉门禁。

3.2：L7 快验契约，L8 留下样板间

fullcheck 真运行时两关不是重复：L7 用隔离换速度，L8 用共享宇宙换覆盖密度。

01 组织模型不同
L7 一场景一 org；L8 一宇宙多 verify。

02 优化目标不同
L7 快、隔离；L8 厚、可调试、可演示。

03 失败资产不同
L7 留小 org；L8 留 session 给 retry / UI 接力。

没有它：接口契约和产品链路互相漏，Gateway 问题拖到联调才爆。

3.3、Stress 证明量大还站得住

Stress 是闸门之后单独长出来的一脉：复用 CLI 宇宙，但不进默认 fullcheck L7/L8。

01 同一套 CLI 宇宙
setup/shared 数据底座复用，只是把油门加大。

02 不挡日常合并
输出 reports/*-stress JSON，给人看 p95 / 基线。

03 四条压力主线
formula / CFD / CQFS / IM cascade 持续扩展。

没有它：功能对了，但大数据一上来才露出容量问题。

02 / OBSERVABILITY

4、Agent 可观测性工程

fullcheck 回答“能不能合”；Loki 和环境契约回答“出问题后，Agent 能不能自己看见”。

01 Loki 可只读查日志
本地 docker/loki + Alloy，8.131 dev 走 skill/script。

02 按路径反查 traceId
path / ERROR / ACCESS 聚合，不再人肉翻文件。

03 环境入口写给 AI
profile 1/2/3、CNB、8.131 写入 docs / AGENTS / skills。

没有它：拷贝日志、截图喂模型，traceId 靠人翻。

5、让Agent自主管理环境流水线

把“启停全栈”写成 Agent 可执行的契约：改完后端自己重启，验收前端自己点亮。

01 后端最小重启
dev:restart:touched 读 diff，只起改到的服务。

02 前端自动托管
agent:ui:open 预检 session/profile，缺 UI 自动 preview。

03 锁与探测防踩脚
check:services / detect:ui-profile 自证，不靠猜端口。

没有它：手工启停、端口猜测，多 Agent 互相踩环境。

5.2、管道和容器也进 Harness

Harness 最后一公里：管道即代码、环境即容器，AI 才能端到端闭环，而不是写完代码等人点部署。

01 管道即代码
.cnb.yml 纯 YAML，触发器、阶段、密钥 imports 都能进 PR。

02 环境即容器
Nacos / MySQL / Redis / Kafka / ZGraph / Loki 可重建。

03 部署不靠人点
YAML 触发部署、重启、种测试组织，Dev 给人验收。

没有它：CI/CD 仍靠人点网页，环境变成少数人的神机。

6、让Agent可以独立验收界面

后端对，不等于界面可用；本层证明真实界面看得见、点得动。

01 Playwright MCP
真实浏览器，点击、填表、截图、看 DOM。

02 Feature map
60+ linkId：功能 → 页面 → setup 说清楚。

03 直达入口
/dev/session + agent:ui:open，禁止手拼 URL。

没有它：API 绿了，界面还要忍受式手工验收。

7、给Agent完整文档上下文

测试证明做对了；文档系统保证做的是该做的，且下次 Agent 还能接着做。

01 设计定真相
docs/design 承接需求与产品意图，改行为先对齐 SSOT。

02 计划定节奏
docs/plan/active 拆任务，完成后归档 archived。

03 评审挡跑偏
docs/review + plan reviews，把方向问题前置。

没有它：上下文搬运靠聊天，换 Agent 就失忆。

8、给Agent完整指导和技能

日产万行之前，先让 Agent 不迷路、不乱来：文档定入口，Skills 定流程，L1 定红线。

01 文档是地图
AGENTS / entry-guide / docs/state 先告诉读什么。

02 Skills 是驱动
碰到 UI、日志、测试、文档，就按 SOP 执行。

03 L1 是内核检查
20+ 静态守卫，违规代码测试前先拦。

没有它：Agent 进仓先迷路，规则靠口头提醒。

9、流程持续优化

关键不是再写一套测试，而是把每次失败反馈固化成可复用、可诊断、可强制的 fullcheck 能力。

01 正确性先收口
L1-L9 分段闸门，把契约、编译、运行时分开挡。

02 稳定性吃掉 flake
L8-retry、稳定性登记册、锁泄漏修复，让偶发失败可诊断。

03 降成本也要强制
path-aware 降低无效运行，CNB stop hook 让同 diff 必过闸。

没有它：Harness 停在脚手架，flake 和慢反馈一直耗人。

04 / FUTURE ROADMAP

下一段：Harness 自己长能力

我们已经把工程交付层跑通；下一步要把并发、协作、反思和记忆做成系统能力。

01 智能体编排
从三 profile 并行，走向可检查的 parent-worker 管理

02 跨智能体工作流
计划、开发、评审、验收、排障分角色协作

03 智能体自我优化
失败轨迹沉淀为 Skill / Playbook，不重复踩坑

04 记忆系统
文件、日志、会话、截图成为可检索长期记忆

先把工程交付层 Harness 跑稳，再进入自我改进层；智能体不是凭空搭楼，而是在稳定地基上继续生长。

LOOP 全球 AI 生态与创新峰会

HARNESS BOUNDARY

后续演进方向

在 Harness 全景里，工作流、验收、权限、状态和运行时被写成一套 Agent 可以执行的系统。Kanban2 这次复盘聚焦最贴近真实交付的一段：让 Agent 跑稳、验住、接进主干。

早期 Agent 框架 · LLM + Memory + Tools + Planning + Action

工程交付层 Harness · Workflow + Evaluation + Permission + State + Runtime

自我改进层 Harness · 智能体编排 + 记忆系统 + 轨迹反思 + Meta-Harness

参考：Lilian Weng, Harness Engineering for Self-Improvement, 2026-07-04

WE DID / NEXT

已做
工程交付层 Harness

未做完
自我改进层 Harness

先把 Agent 稳定交付做成系统，
再让系统从轨迹里继续长能力。

ROADMAP

目录

01	日产万行不是梦	Commits, churn, and pressure
02	Harness 建设之路	Agent OS, fullcheck, CLI E2E, skills, hooks
03	使用心得	When platform work starts compounding
04	落地建议	Reusable guardrails for high throughput

LOOP
全球 AI 生态与创新峰会

BUILD THE
HARNESS ROAD

真实用法：先控计划，再放手执行

流程不是让 AI 直接开写，而是先把意图、计划和评审固化，再交给 Agent 在 Harness 里跑。

01 制定计划

图 3：先让模型理解意图和边界。

02 生成计划

图 2：把方向拆成可执行 PR 计划。

03 评审计划

图 1：计划落盘后进入评审修订。

04 放手执行

图 4：计划更新后让 Agent 开始改代码。

一小段意图 → 低成本模型出计划 → 强模型评审 → Agent 在开发机执行

坚决重构，不留债务

计划评审阶段一发现架构异味，就先拆、先改、先收口；不把债务带进下一轮并发。

01 **233 次 refactor**
两个月显式重构提交，约 8.3% 同期提交

02 **195 次业务向**
扣除 docs / archive 后，仍保持 3.2 次/天

03 **强审看架构**
评审者要能闻到债务苗头，不只看排期

口径：2026-05-22 至 2026-07-22，非 merge，subject 以 refactor 开头。

意图进计划，执行过闸门

Harness 不是让 AI 裸写代码，而是把意图、计划、评审、执行、验收都压进同一条主干流程。

01 计划必须落盘
Auto 产出 active plan，强模型评审进 reviews。

02 执行机负责关门
三套 CNB profile 并行，fullcheck(+UI) 才算做完。

03 Dev 交给人验收
带 fullcheck、截图、计划链接和日志线索看结果包。

同一套闸门，两种验收时机

Harness 建好之后，还要决定何时关门：云上用时间换 Agent 闭环，本机用人换速度。

01 CNB 强制关门
改业务跑 fullcheck；UI 改动再接 Playwright followup。

02 本机默认不挡路
cnb-mode=false 时 hook 跳过，人先点 UI、先联调。

03 不是两套代码
同一套 Harness，只是验收调度策略不同。

ROADMAP

目录

01	日产万行不是梦	Commits, churn, and pressure
02	Harness 建设之路	Agent OS, fullcheck, CLI E2E, skills, hooks
03	使用心得	When platform work starts compounding
04	落地建议	Reusable guardrails for high throughput

LOOP
全球 AI 生态与创新峰会

BUILD THE
HARNESS ROAD

智能体是长出来的：先让人跑稳，再让 Agent 提效

不是上来就造专用智能体。先在一个真实但可控的系统里，把 Harness 跑稳，再让 Skill 和智能体自然沉淀出来。

01 选中等复杂度系统
太简单看不出效果，太复杂周期太长

02 找业务 + 架构 Pair
一个懂业务，一个架构强；一人兼具更好

03 先补端到端守护
守住基本能力，再开始重构提速

没有 Harness 根基，智能体建设很容易变成空中楼阁。

TGO 鲲鹏會

Q&A
讨论

LOOP 全球 AI 生态与创新峰会

先证明高吞吐，再拆 Harness；最后把用法和清单带回自己的团队。