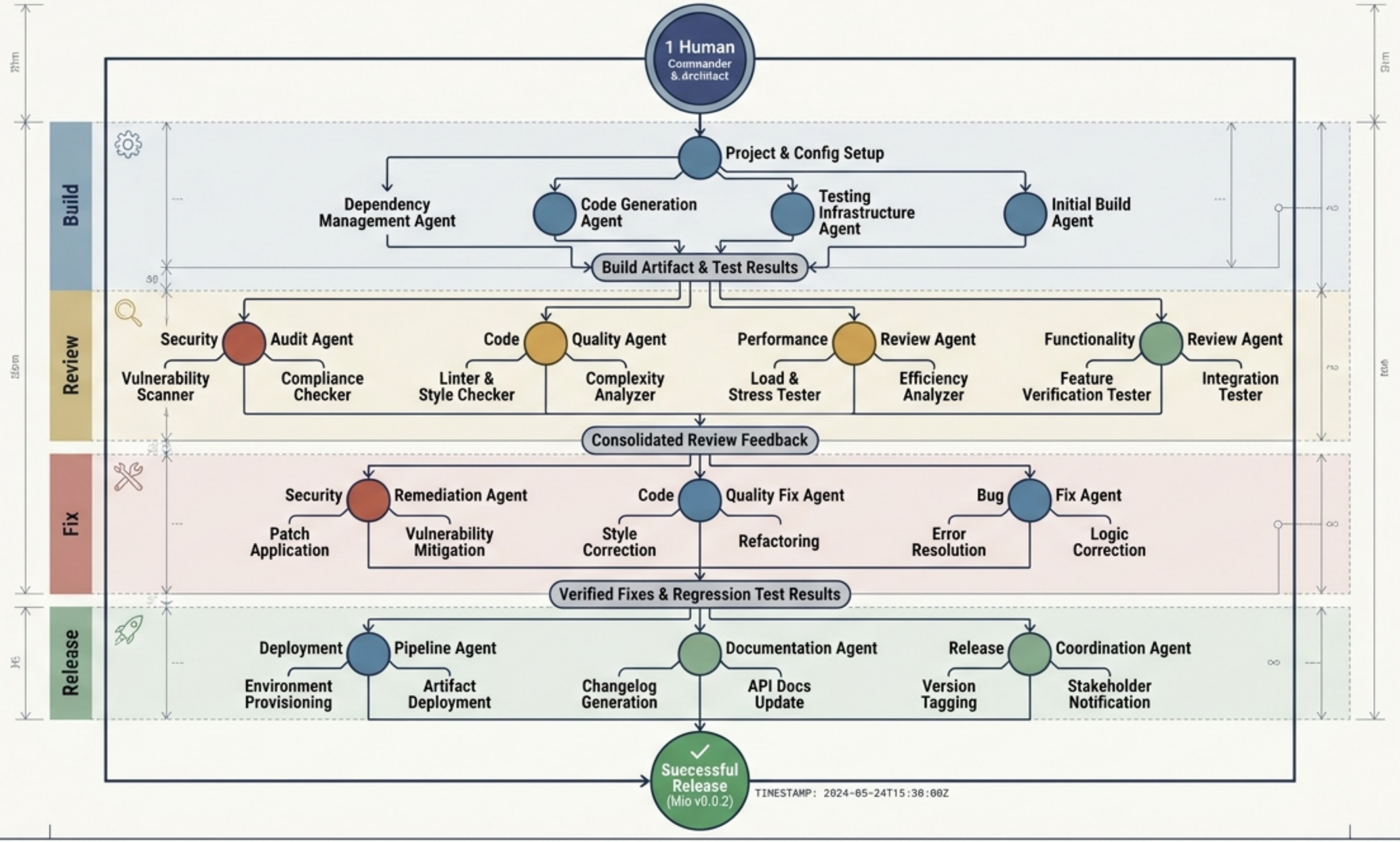
从“AI助手”到“AI工程团队”：多智能体并发代码审查与交付剧本

一份基于 `Mio v0.0.2` 的实战复盘，探索由 1 人指挥、14 个独立 AI 智能体并行协作的下一代软件开发生命周期（SDLC）。



NotebookLM

代码能跑，绝不等于系统扛得住

Five concurrent agents produced 81 commits spanning a three-tier architecture in the last build. Code ran successfully, type checks passed unanimously.
But in the baseline scan after crash recovery, a hidden crisis ignored by the test suite emerged. We enter the other side of the same workflow: Review, Fix, and Release.

BUILD SUCCESS: CALM & ORDERLY

BUILD PASSED

100%

PROGRESS: 100% COMPLETE | DURATION: 4m 32s

```
user@dev-environment:~/project/build$
> running pipeline: pre-commit hooks...
> agents: 5 active nodes (agent-01 to agent-05)
> architecture: 3-tier (frontend, backend, data-layer)
> commits: 81 commits spanned

[SUCCESS] Commit processing complete. (81/81)

> running static analysis: type-checker...
[SUCCESS] Type Checking: 0 Type Errors found in 894 files.

[SUCCESS] All systems operational. Ready for deployment.
```

81 COMMITS

0 TYPE ERRORS

HIDDEN CRISIS: TEST BLIND SPOT

```
export const config = {
  ...
  endpoint: 'https://api.service.com/v1',
  timeout: 5000,
  ...

const API_KEY = "sk-1234567890abcdef1234567890abcdef";
// TODO: Move to environment variable! Hardcoded secret.

};
```

CRITICAL SECURITY VULNERABILITY DETECTED

Hardcoded API Key exposed in production build artifacts. Ignored by standard test suite.

FILE: /src/config/service_config.ts (Line 42)
SEVERITY: HIGH | CWE-798: Use of Hard-coded Credentials

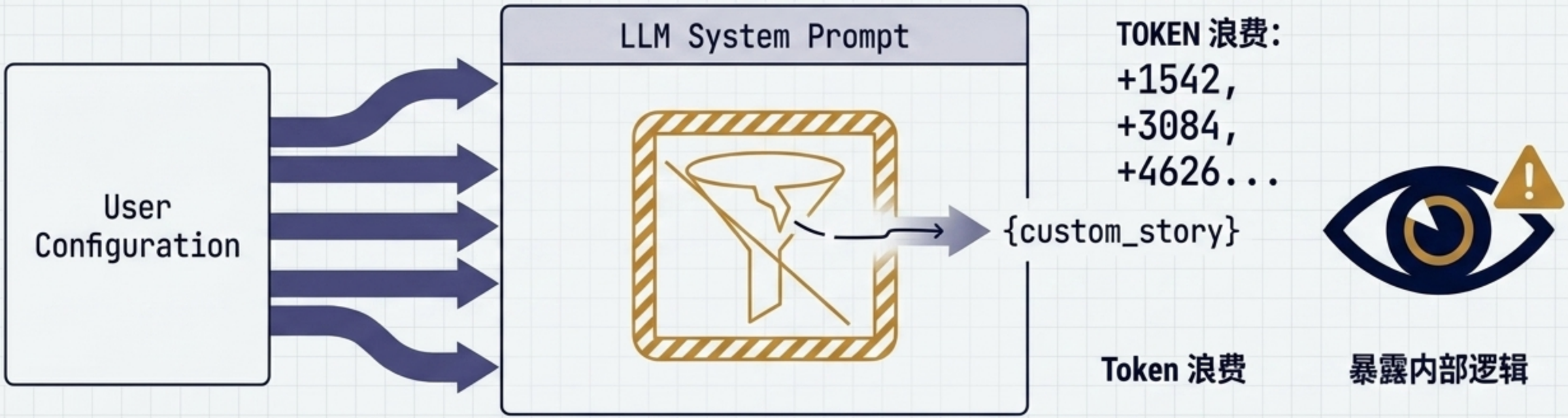
NotebookLM

触发全盘审查的导火索：占位符泄漏

人格配置文件中本该在 onboarding 阶段被替换的**死模板 {custom_story}** 泄漏到了最终发给大模型的 System Prompt 中。

后果：

- 1. Token **浪费**：模型照常运行（代码"能跑"），但每一次交互都在白白消耗算力。
- 2. **暴露内部逻辑**：一旦有人审查 Prompt，未经处理的占位符将导致严重的尴尬与专业度受损。



NotebookLM

启动并发审查：四个专业人格的诞生

COMMAND EXECUTION

```
> ok do a full review with agent team of the implementations
```

AGENT 1: MEDIA EXPERT

SPECIALTY: Media, Assets, Content Delivery

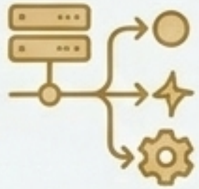
Focused on media file integrity, format compliance, and delivery pathways. Parallel scanning of asset libraries and content pipelines.

AGENT 2: ONBOARDING GUIDE

SPECIALTY: User Flows, Authentication, Onboarding

Dedicated to reviewing user registration, login flows, and first-time experience logic. Ensures smooth onboarding and identity verification.

AGENT 3: PIPELINE ARCHITECT



SPECIALTY: CI/CD, Infrastructure, Deployment

Examines build scripts, deployment configurations, and infrastructure-as-code. Focuses on system stability and deployment readiness.

AGENT 4: SECURITY AUDITOR

SPECIALTY: Vulnerabilities, Permissions, Data Privacy

Scans for security flaws, hardcoded secrets, and permission issues. Prioritizes risk assessment and compliance with security standards.

不使用单一的"全能"AI 审查代码。我们同时部署了四个具有专门知识域的独立智能体 (Agents) 。各自圈定视界，并行扫描代码库。专业化使得安全专家和功能审查员在看同一份文件时，能看到完全不同的切面。

NotebookLM

为什么需要专业化？通用审查与专业审计的视角差异

智能体角色	核心关注领域	发现的典型漏洞	审查范式
Media 审查员	Vision，Process，Selfie，引用图片，文件下载	引用图片缓存无上限	资源与边界约束
Pipeline 审查员	Router，Index，System-prompt，Agent 循环	成本归因前置，导致追踪失败	数据流与状态正确性
Onboarding 审查员	状态机，预设配置，Schema，命令	Onboarding 过程中的竞态条件	逻辑与用户体验流
Security 审计员	API Key，输入验证，注入，SSRF	目录穿越，Bot Token 泄漏	攻击面探测与防御

NotebookLM

五分钟内的并发审计结果：拦截发布

四个审查报告并行返回。没有合并冲突，只有冷酷的数据：

Pipeline

2 CRITICAL, 4 HIGH, 6 MEDIUM

BLOCK

Security

2 CRITICAL, 4 HIGH, 5 MEDIUM

BLOCK

Media

0 CRITICAL, 4 HIGH, 10 MEDIUM

WARNING

Onboarding

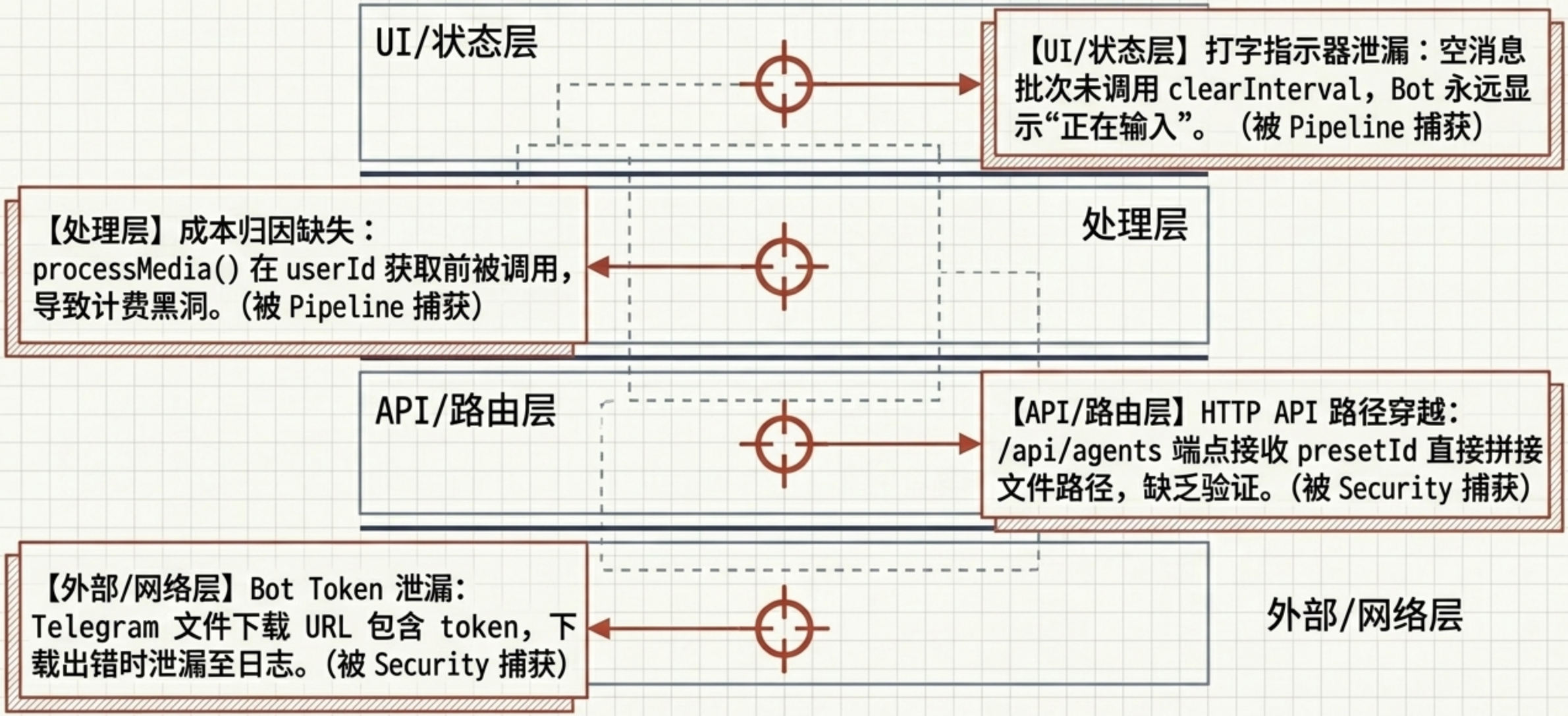
0 CRITICAL, 4 HIGH, 7+ MEDIUM

WARNING

结论：2 个 BLOCK 级阻碍，
版本冻结，进入并行修复阶段。

NotebookLM

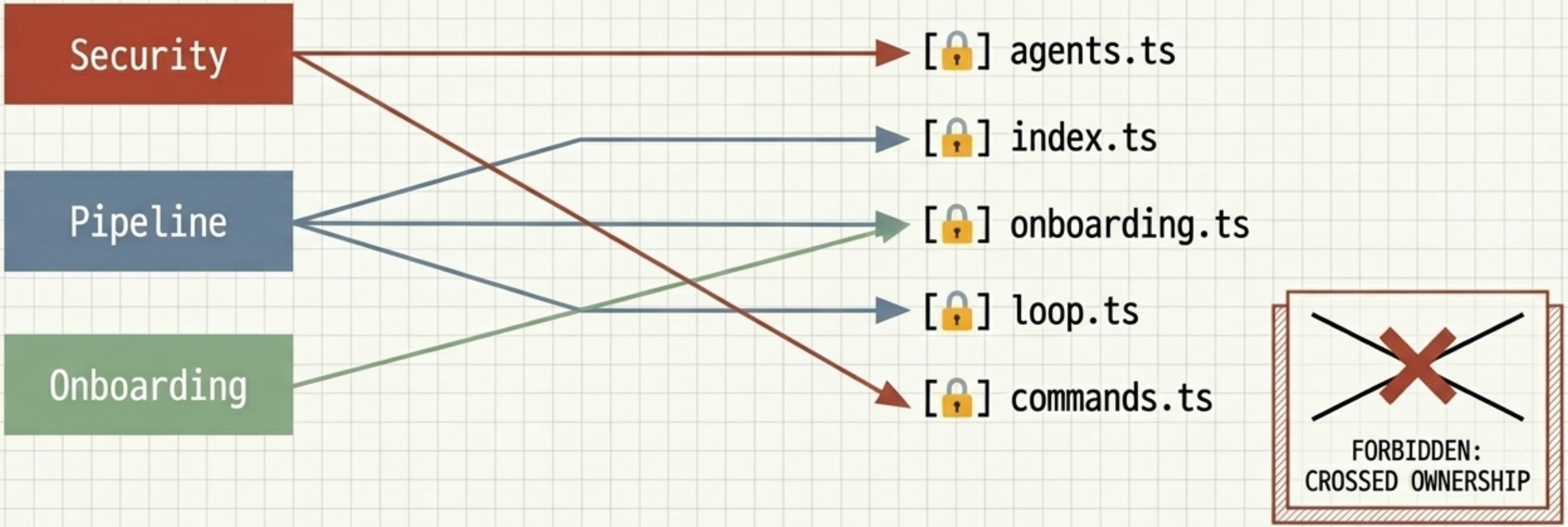
四个致命缺陷的架构级映射



NotebookLM

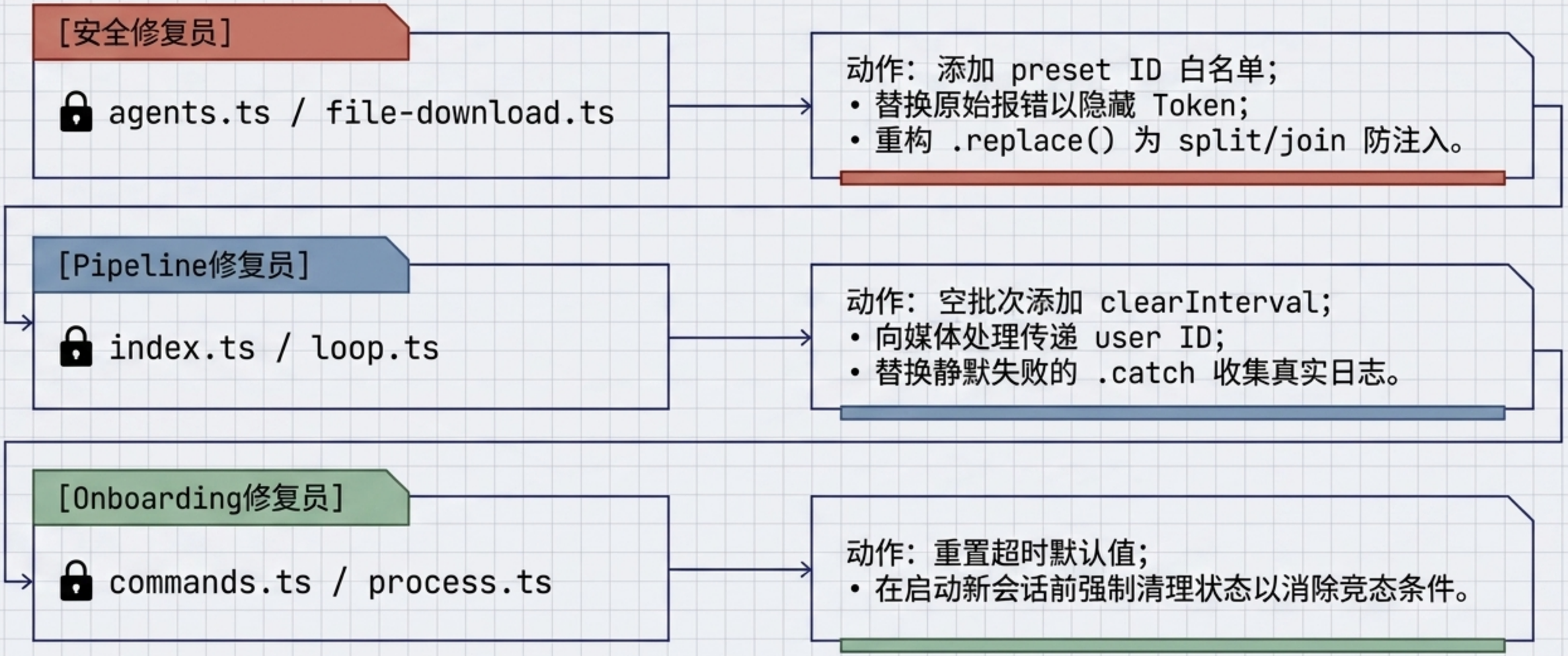
并发修复的铁律：严格的文件绝对所有权

没有两个 Agent 编辑同一个文件。
这是多智能体并行的核心机制。任务不按 Issue 分配，而是按文件所有权分配。如果两个 Issue 涉及同一个文件，它们必须被打包交给同一个修复员。文件锁定是防止撕裂和代码合并冲突的唯一解。



NotebookLM

三线并行的修复员：无冲突作业网

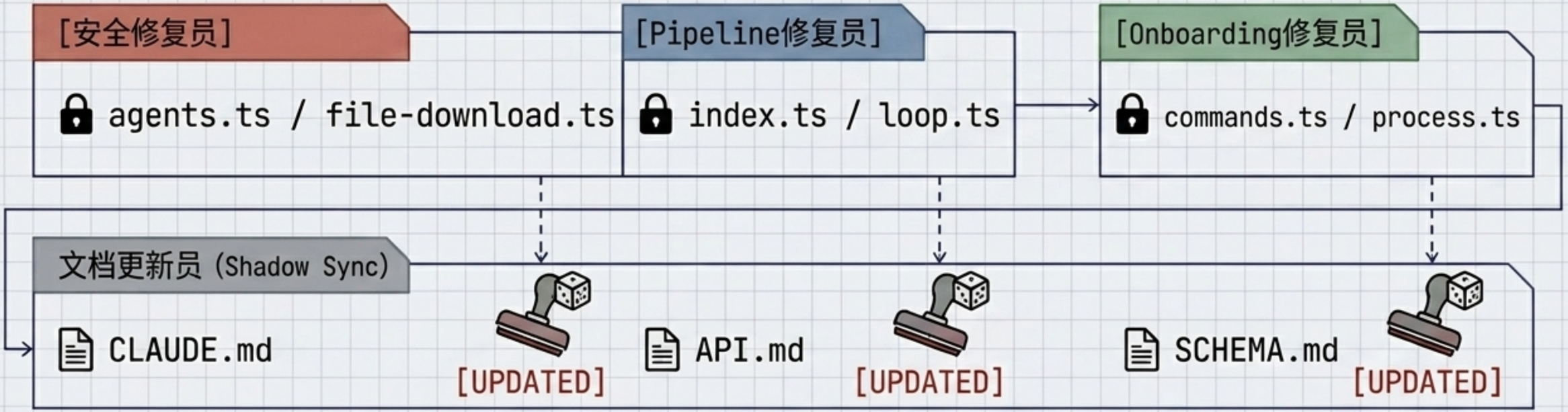


NotebookLM

消除文档漂移：第四个智能体的影子同步

代码变更发生的同一时刻，第四个“文档更新员”正在后台同步运行。它全面更新了 TECHNICAL.md，DATA-FLOWS.md，以及 CONVENTIONS.md。

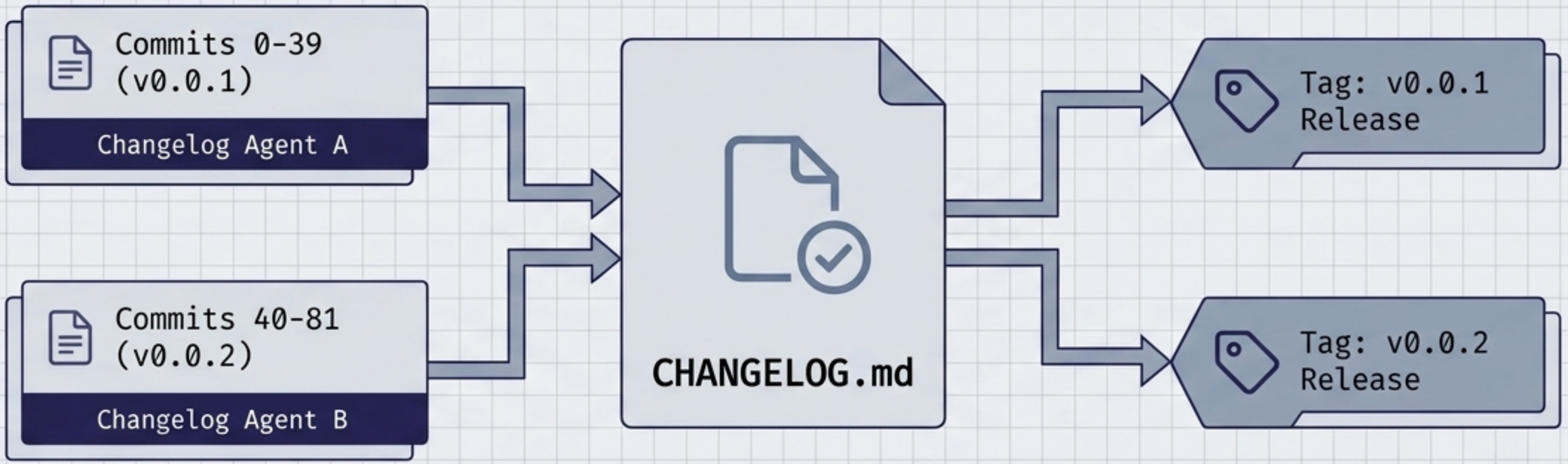
定律：文档永远不要等“以后”再写。在 AI 团队中，并行运行一个文档更新 agent 几乎没有边际成本，这从根本上终结了“文档漂移（Document Drift）”的业界顽疾。



NotebookLM

仪式性工作自动化：把机械动作交给机器

所有修复提交后，人类只需下达一条指令：let's also add a changelog thing. 两个 Changelog Agent 并行启动：一个负责前 39 个基础构建 Commit，另一个负责涵盖审查与修复的后 81 个 Commit。



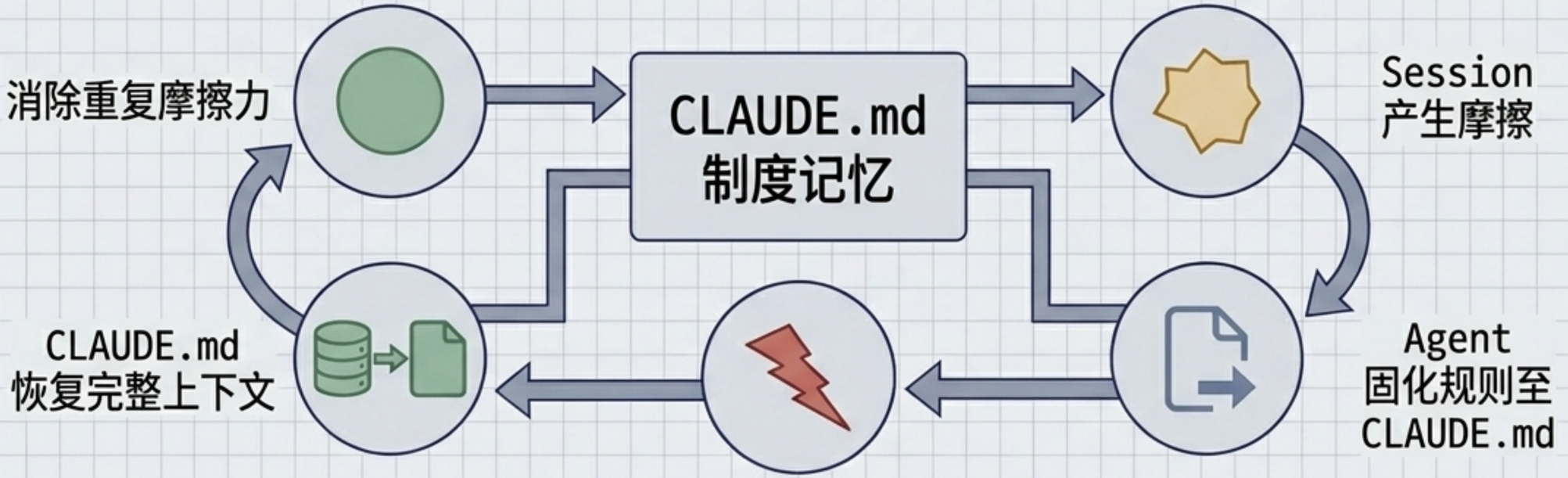
发布流：生成日志 -> 整合汇总 -> 打下两个 Tag -> 创建两个 GitHub Release。全流程人类零干预，只需旁观。

NotebookLM

跨越崩溃的系统记忆：CLAUDE.md 飞轮

将复盘经验（如“发布与版本控制工作流”、“永远使用文档更新子模块”）硬编码写入项目根目录的 CLAUDE.md。

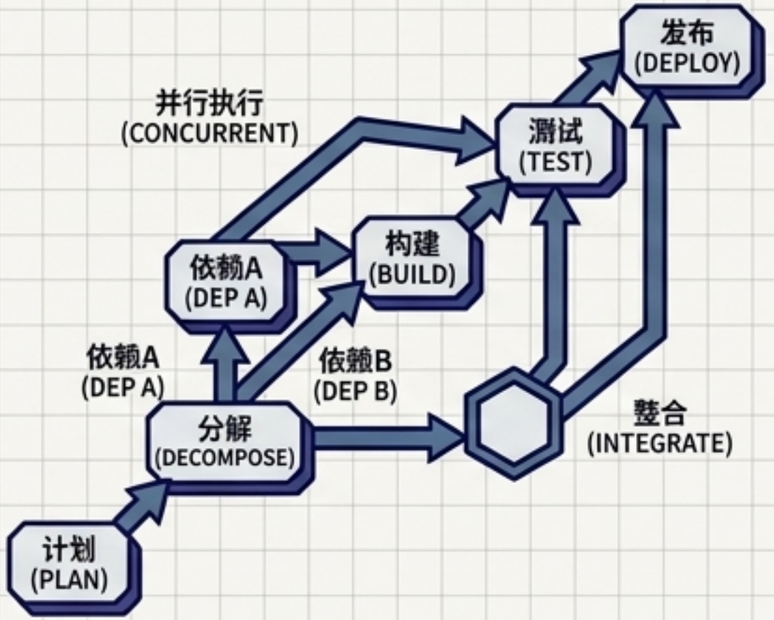
这不仅仅是笔记，这是项目的“制度记忆”。它能跨越 Session、跨越上下文压缩、跨越系统崩溃实现持久化。写入得越多，系统重置时丢失的上下文就越少，摩擦力呈指数级下降。



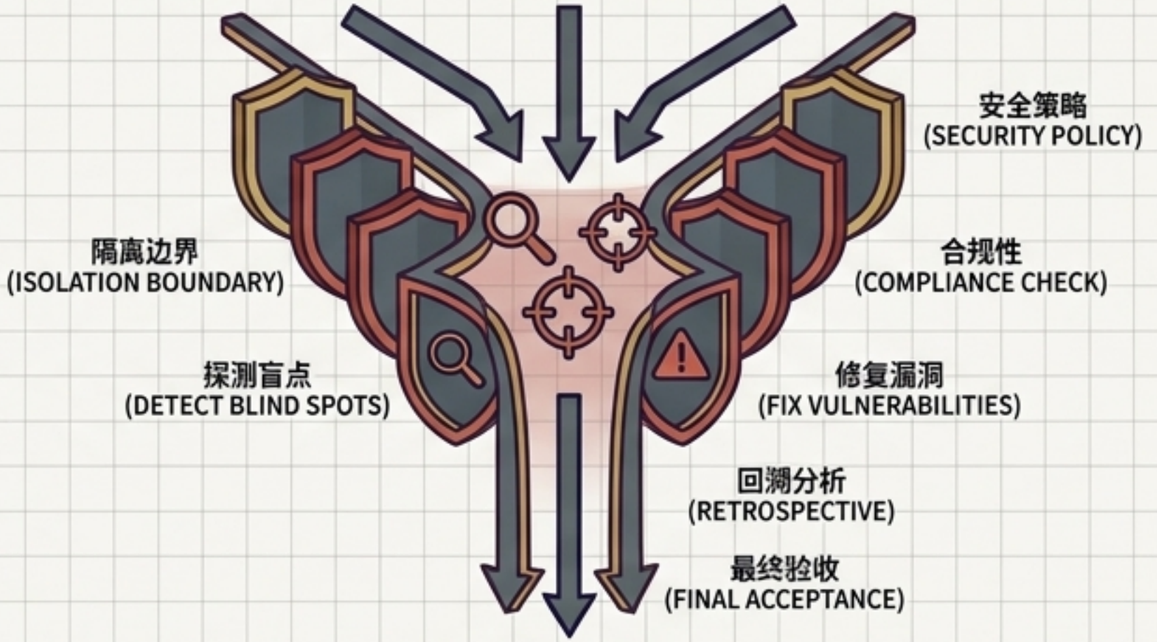
NotebookLM

同一个工作流的两面：创造与对抗

构建期（Building）



审查期（Reviewing）



创造性的，向前看的	核心性质	对抗性的，回溯性的
分解计划，调度依赖，DAG 驱动	执行逻辑	划定边界，探测盲点，隔离修复
架构师与指挥官	人类角色	最终决策者与验收官

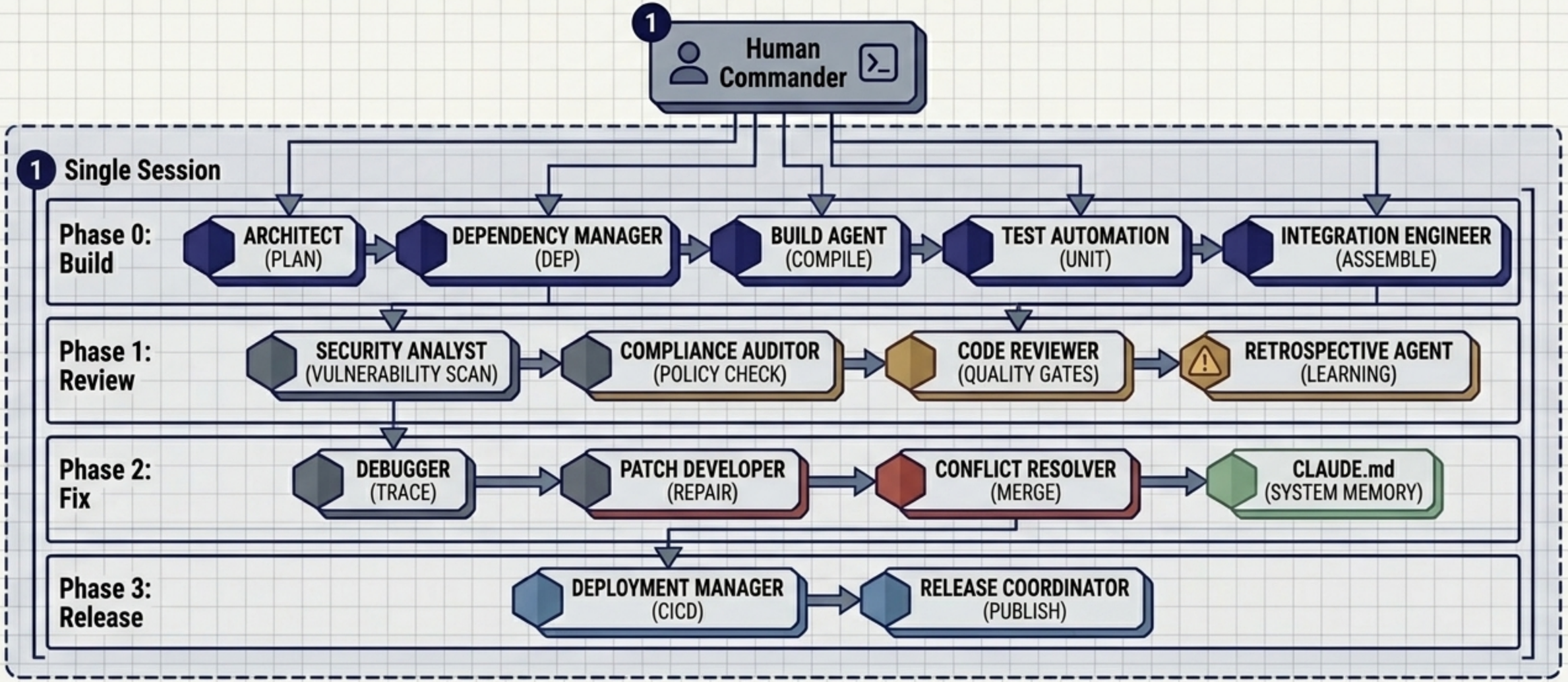
共同点：高度依赖专业化分工，高度依赖并发执行与文件所有权

NotebookLM

智能体协同全景：单会话驱动十四人团队

不要只用 AI 来编写代码，要用 AI 工程团队来交付系统。

规划构建、专业审查、无冲突修复、自动发布、制度记忆沉淀。这不是未来的构想，而是我们在单一会话中已经落地的现实。



NotebookLM

AI 原生工程体系的五大黄金法则

AI-Native SDLC

［特殊化］

审查员必须专业分化：通用审查只看正确性，专项审计才能锁定致命攻击面。

［隔离性］

严格约束文件所有权：绝不不允许两个修复员触碰同一文件，职责按文件而非 Issue 划分。

［同步性］

文档更新与代码修复并行：启用影子智能体同步维护架架构文档，终结文档漂移。

［自动化］

剥离一切仪式性工作：打标签、写 Changelog、发版本等机械环节，全部交由 AI 闭环执行。

［持久化］

流程改进立刻落实于纸面：将摩擦点固化至 CLAUDE.md，实现跨会话复利。

WORKFLOW: REVIEW, FIX, RELEASE | STAGE: CONCURRENT REPAIR PHASE | STATUS: IN PROGRESS | SYSTEM HEALTH: RECOVERING

NotebookLM