PALANTIR SERIES 04 — ONTOLOGY → FOUNDRY → AIP

医院里的 AIP
AI 成本降近 70%

前三期讲本体论与 Foundry，这一期用坦帕综合医院案例，讲清 AIP 如何降成本、减调用，还让专家更满意

本体论
让 AI 理解企业里的人、事、关系与规则

ONTOLOGY

FOUNDRY
把数据、流程和应用连成可运行的生产系统

DATA + OPERATIONS

AIP
让 AI 在这套底座上运行，并持续评估和优化

AI + EVOLVE



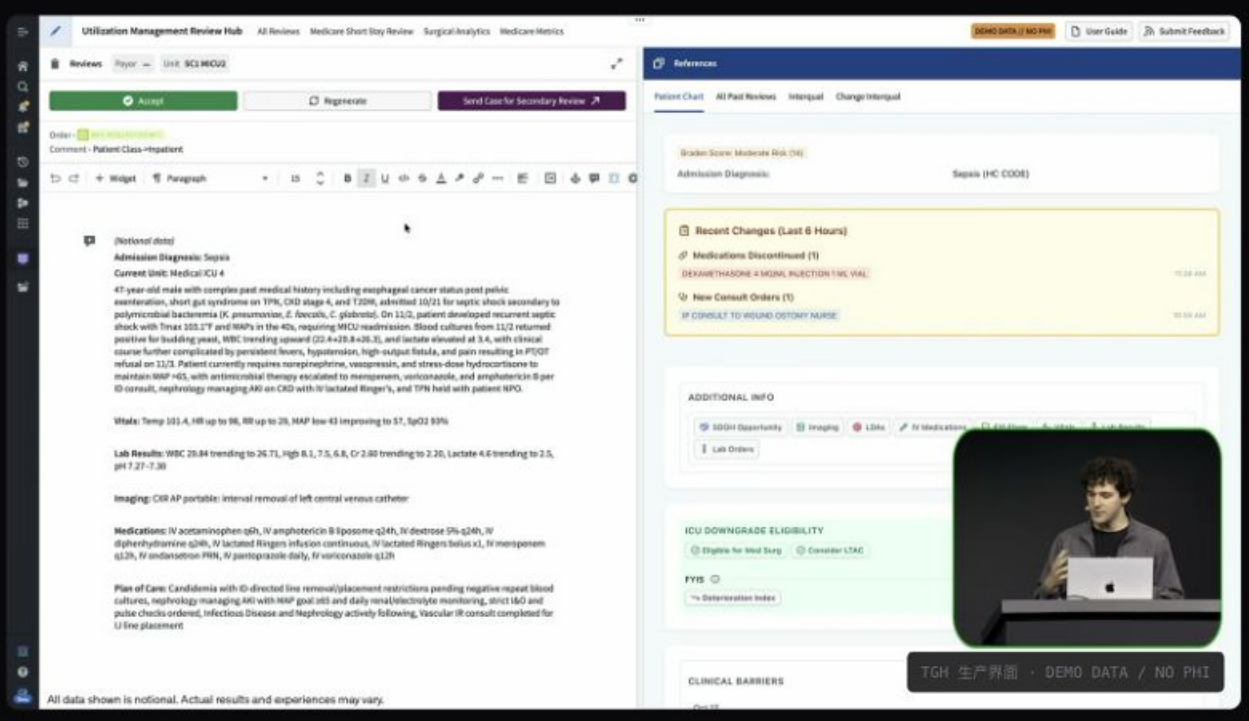
David Zirh, MD, MBA（Director, Physician Advisor Services, TGH）+ Colton Rusch（Forward Deployed Engineer, Palantir）

他们做出来的结果

70% ↓	84% ↓	90%
AI推理成本	GPT调用次数	专家偏好优化后输出

口径：70% 指 AI 推理与算力成本，非医院整体运营成本。
来源：Palantir DevCon 6 官方分享，非第三方独立审计。
场景：医疗资源利用审核（Utilization Review）— 分析病历判断住院和治疗合理性，直接关系保险报销。

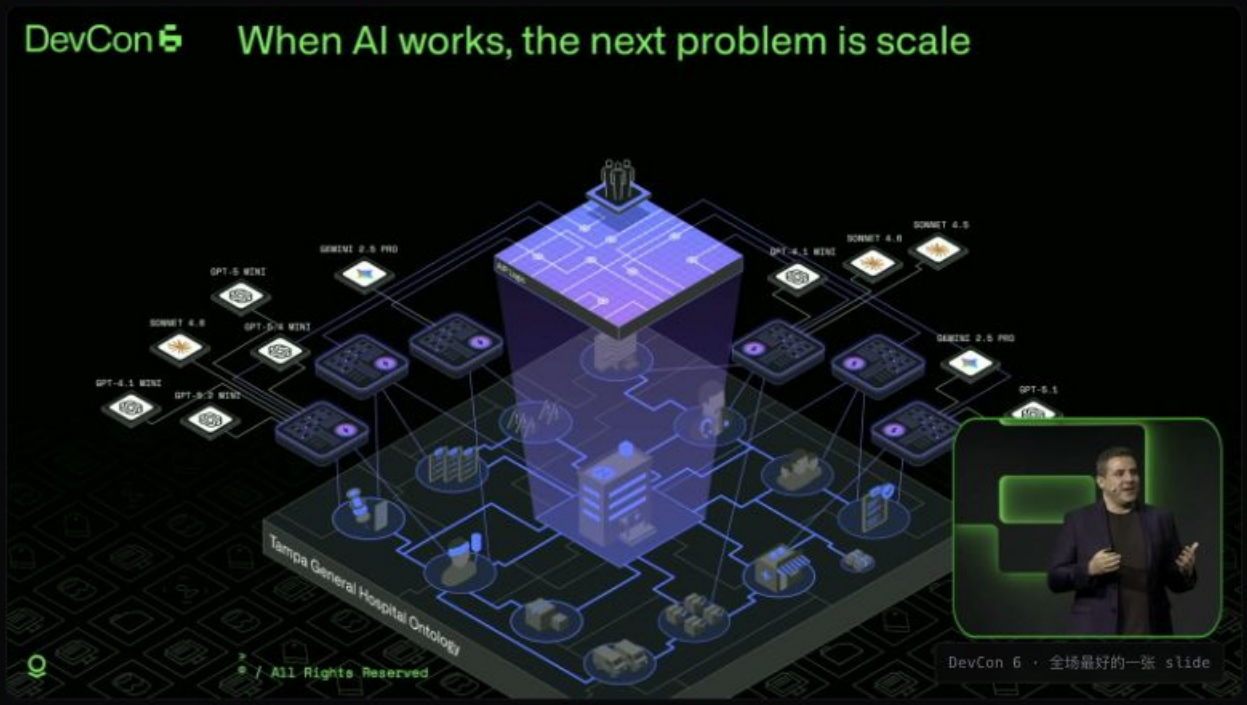
> 这不是简单把大模型换成小模型，而是把"找浪费、守质量、再上线"做成了一套优化流程。



真实生产界面：脓毒症病例的 AI 生成审查草稿、Braden Score、ICU 降级资格判断

WHY COST EXPLODES

AI 跑通之后，成本为什么会失控?



一个真实工作流，周围是多个候选模型。每增加一个环节、模型、提示词或工具，组合空间都会迅速膨胀

组合空间大到人工穷举不完。

工作流环节 × 模型 × 提示词 × 工具调用 × 架构变体。团队很快就无法靠人工把所有组合试一遍。

所以现实中大家怎么做?

AIP Evolve 要解决的，就是把"选哪个模型、改哪段提示词、哪次调用可以删"从拍脑袋变成可验证的自动寻优。

Palantir 给它的定位是 Automated AI Value Maximization — 自动化的 AI 价值最大化。

HOW IT WORKS

四步只回答四个问题

优化什么、质量红线是什么、允许改什么、最后由谁批准。点标签查看原始界面。

01 GOAL　02 VALIDATION STRATEGY　03 AGENT CONSTRAINTS　04 REVIEW

选目标

Target 选 `AIP Logic` — 就是要优化的那个 AI 工作流。

Goal 选 `Optimize cost` — 优化成本。

OUTPUT

它给的不是答案，而是可审核的权衡

同样是降成本，两份真实提案得到完全不同的结论。

VERDICT: SAFE TO PROCEED · CONFIDENCE: HIGH

Cut compute cost 97% by swapping Block A from GPT-5.1 to GPT-5.4 nano

▼97%	▼69%	▲+7pp
Compute cost	Latency	Quality
306.9 → 7.6 compute-sec	14.6s → 4.5s	28/30 → 30/30

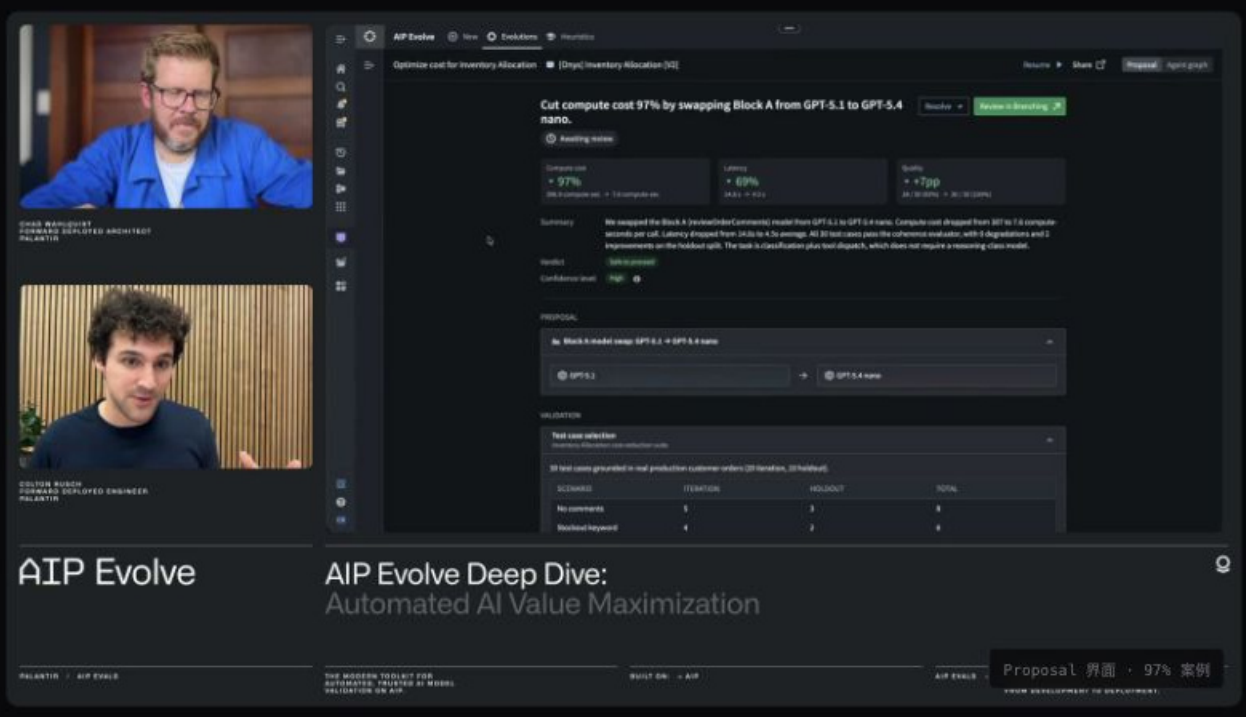
它给的判断依据是关键：这个任务是"分类加工具调度"，不需要推理级模型。
验证明细：30 个真实生产用例，20 个用于迭代，10 个 holdout。

VERDICT: REVIEW RECOMMENDED · CONFIDENCE: MEDIUM

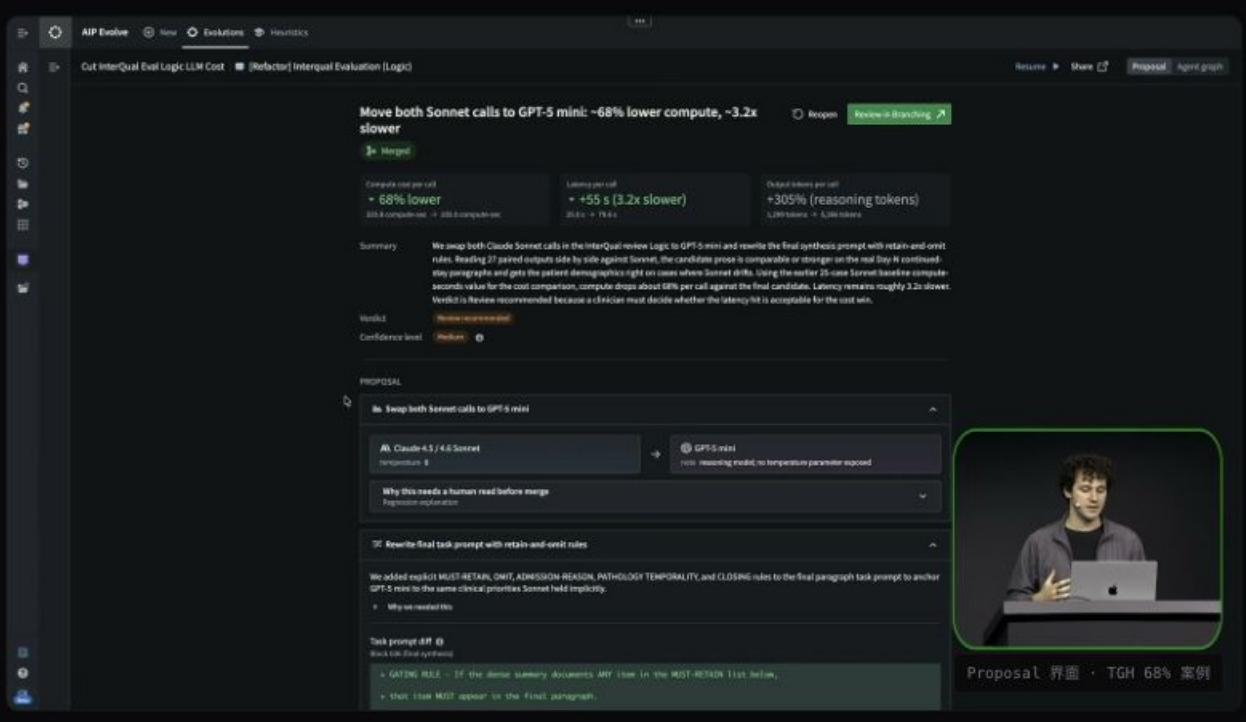
Move both Sonnet calls to GPT-5 mini: ~68% lower compute, ~3.2x slower

▼68%	3.2×	Medium
Compute cost	Slower	Confidence

坦帕医院案例里，省 68% 的代价是慢 3.2 倍。
这个交易划不划算取决于场景 — 批量夜间处理没问题，实时交互就不行。



Proposal 界面 · 97% 案例



Proposal 界面 · TGH 68% 案例

不存在脱离场景的最优模型，只有满足业务约束的最优组合。

夜间批处理可能接受慢三倍；实时交互可能不能。系统把成本、速度、质量和置信度摊开，最后由人决定。

compute-second 是什么单位?

THREE COST LEVERS

不用 Palantir，也能照着做

先把推理成本拆开：总调用次数 × 单次 token 数 × 模型单价。优化就按这个顺序做。

调用次数 × 单次 token × 模型单价 = 推理成本

失败重试、agent 自我检查和多轮纠错，最终都会回到这三个旋钮上

① 先减少调用次数 — 收益最大

> 换模型只是让一次调用更便宜；删掉调用，是让这笔钱彻底消失。
>
> AIP Evolve 在案例中发现了已有的结构化 Ontology 数据，因此直接消除了两次 LLM 调用。

每个工作流节点只问四个问题 （点击展开）

01	答案有明确规则吗?	用代码
02	结构化数据里已有答案吗?	用查询
03	只是抽取、分类或匹配吗?	小模型 / 传统 ML
04	需要理解模糊语言、推理和生成吗?	才用大模型

② 再减少单次 token

DEDUP

删除重复上下文

同一条规则不要在 system prompt 和 user prompt 里重复出现。

RETRIEVE

先检索，再提供

不要把整个知识库塞进上下文，只给当前任务真正相关的片段。

CONSTRAIN

限制输出形状

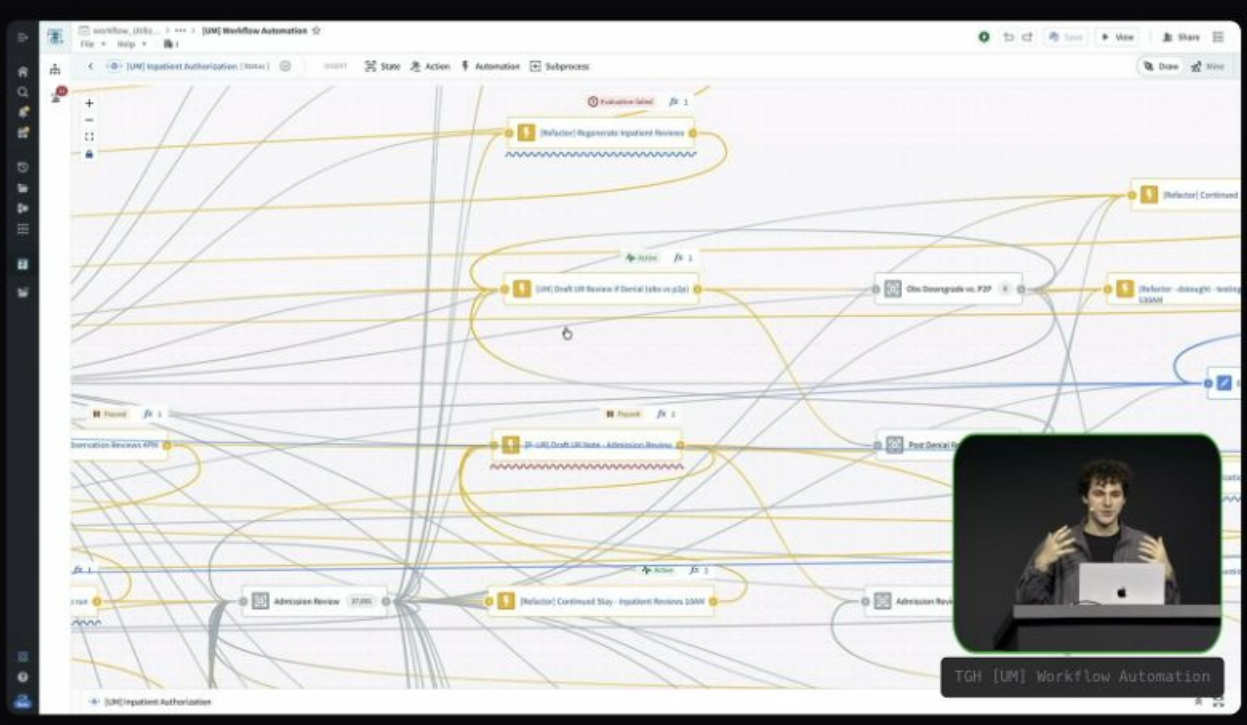
只需要字段或 JSON 时，不要让模型生成一大段解释。

> 固定且高频复用的前缀可以做 prompt caching，但先看命中率和真实账单，不要只看理论折扣。

③ 最后做模型路由

> 不是“小模型优先”，而是“任务分级”。
>
> 字段抽取、简单分类优先测试小模型；复杂推理、长文档和低容错任务再用强模型。每类任务都要有质量下限、延迟上限和成本目标。

> 三个旋钮外面必须有护栏。
>
> Tracing 告诉你钱花在哪；Evals 告诉你改完有没有退化；holdout 防止只对测试题优化；人工复核守住高风险场景。



生产工作流的真实复杂度：节点越密，越需要逐节点判断"这一步真的需要 AI 吗？"

THE GUARDRAIL

Evals 是降本的护栏，不是附属功能

AIP Evals

"我的 AI 系统表现如何？"

- 定义测试用例
- 选评估器
- 跑实验
- 出结果对比表

尺子

AIP Evolve

"怎么让它更好？"

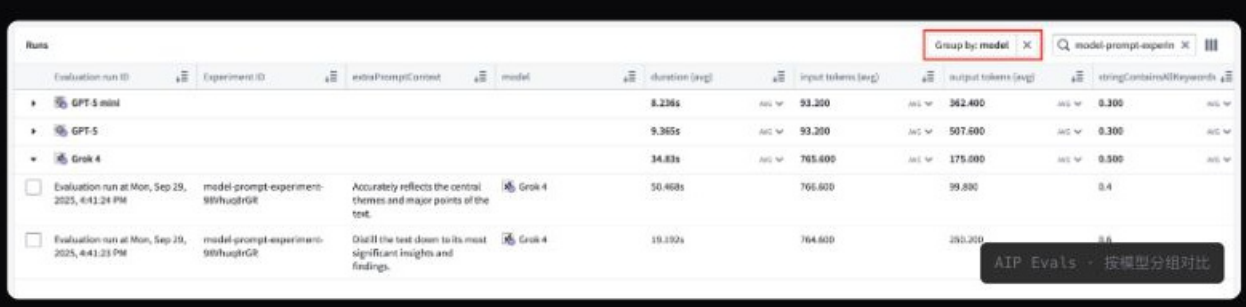
- 配置优化目标和约束
- 交给 AI FDE 执行
- 自动迭代寻优
- 出带判断的提案

拿着尺子做改进的人

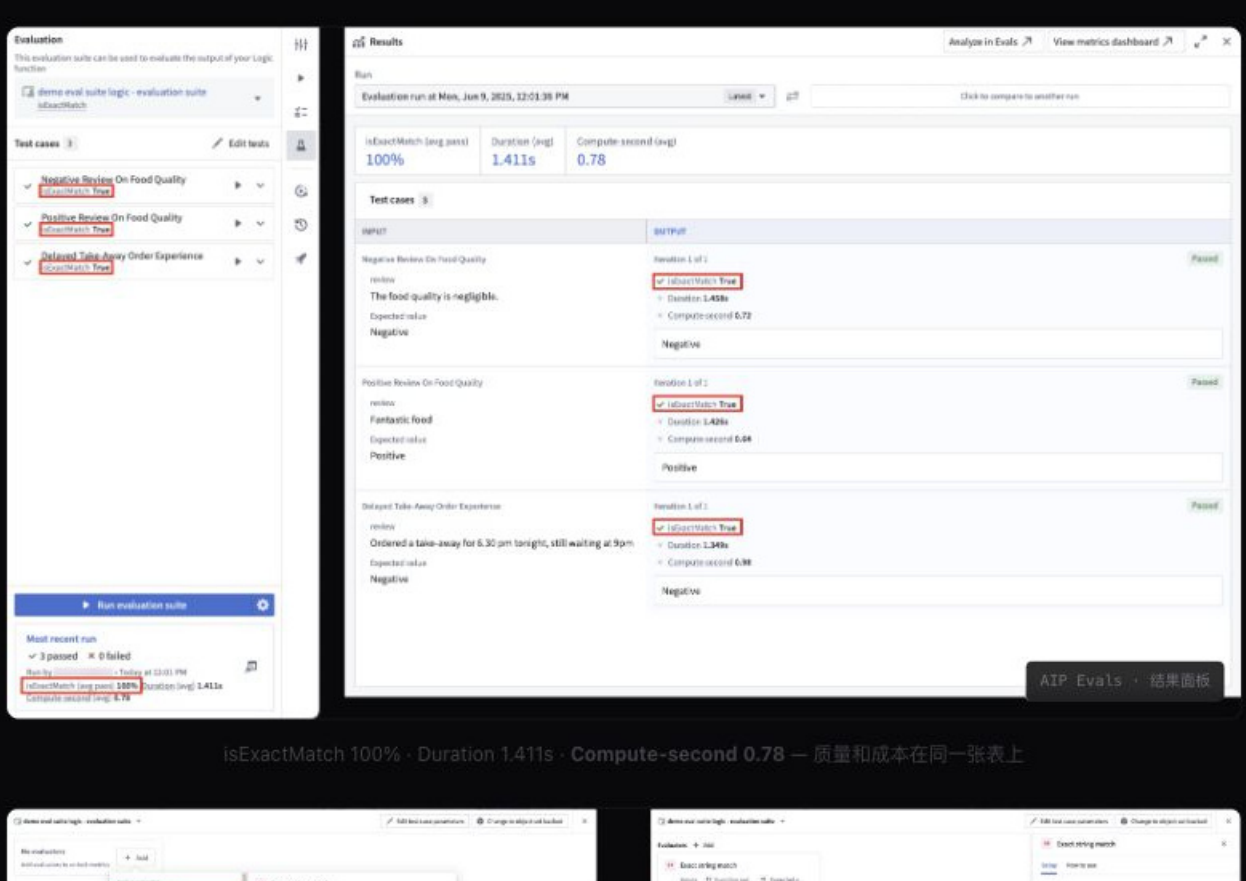
两者的关系写在那段发给 AI FDE 的 prompt 里：

"用 AIP Evals 和你生成的测试集来验证没有出现有意义的退化。"

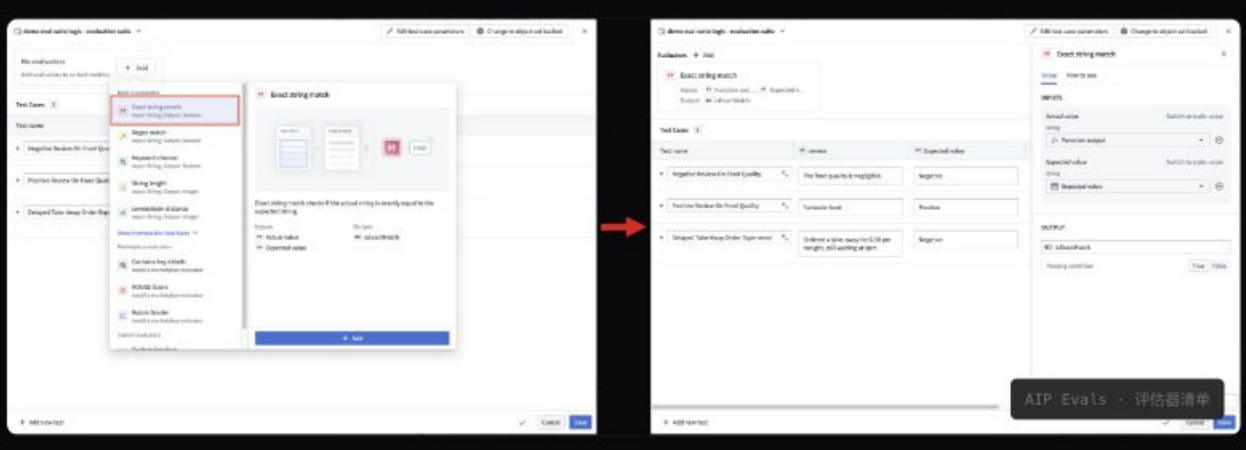
Evolve 在优化过程中调用 Evals 做验证。没有评估的降本是在赌，有评估的降本才是工程。



GPT-5 mini 8.236s / GPT-5 9.365s / Grok 4 34.83s — 含 input/output token 均值，这才是选型该有的依据



isExactMatch 100% · Duration 1.411s · **Compute-second 0.78** — 质量和成本在同一张表上



截图中的旧界面包含精确匹配、正则、关键词、Levenshtein、ROUGE 与 Rubric Grader；当前官方已用更直接的 LLM-as-a-judge 取代 Rubric Grader

至少守住三层验证

DETERMINISTIC

确定性指标

精确匹配、正则、范围、结构校验，适合有明确答案的任务。

SEMANTIC

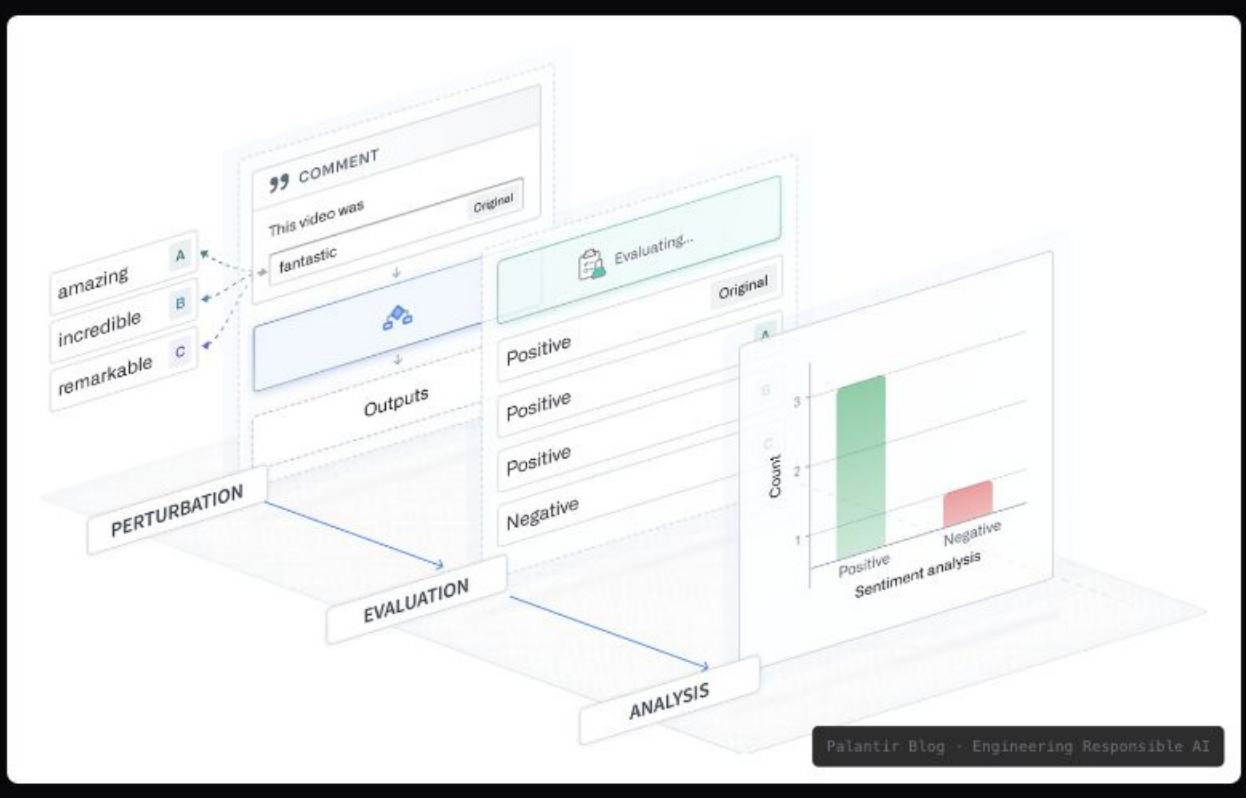
语义与稳定性

LLM-as-a-judge 加扰动测试，检查意思、质量和输入变化下的稳定性。

EXPERT

人工专家基准

高风险输出最终要让业务专家盲测，坦帕的 90% 偏好就来自这一层。



扰动测试三段式：Perturbation → Evaluation → Analysis

ACTIONABLE

明天就能开始的四步

先找到最贵的三个任务，再小范围验证；不要一上来重构所有流程。

01

接入 tracing，按任务拆账

02

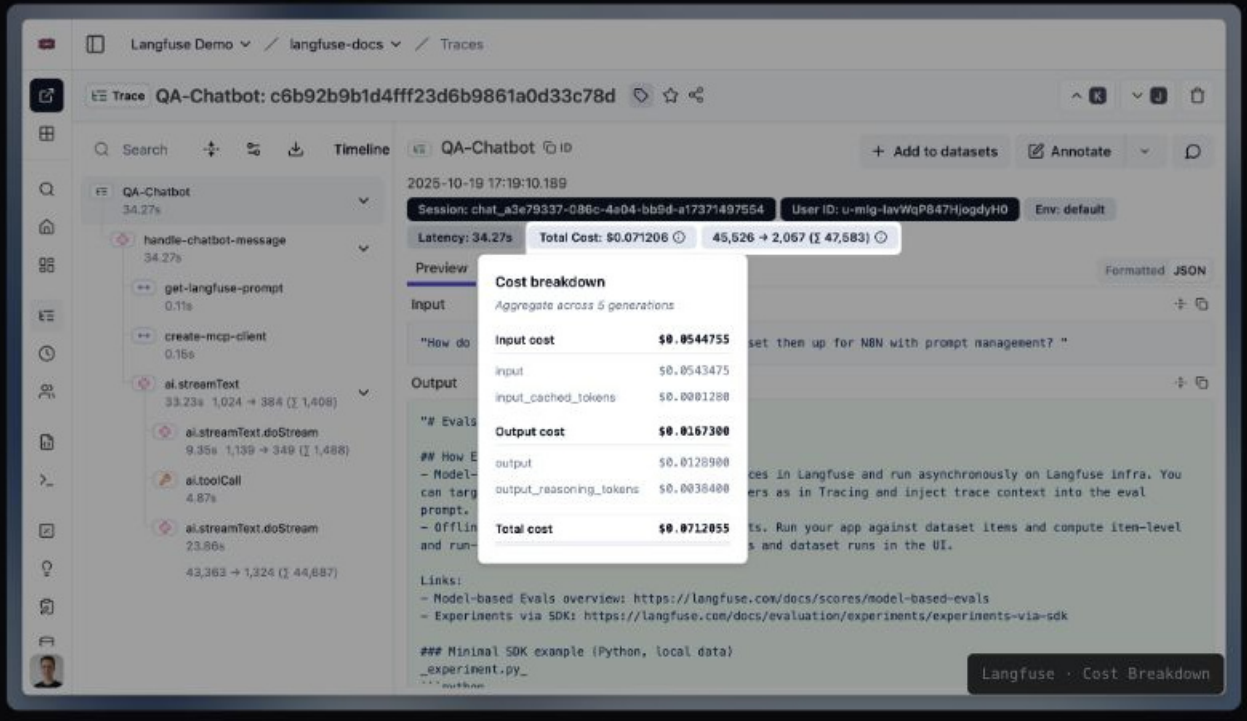
锁定成本最高的三个任务

03

先定红线，再动三个旋钮

04

小流量上线，继续看生产指标



input / output / cached / reasoning tokens 逐项拆分，才能知道钱究竟花在哪个环节

TAKEAWAY

AI 降本，最终只看这四件事

成本侧

少 × 短 × 准

更少的调用
更短的上下文与输出
更匹配任务难度的模型

同时

质量侧

可证明

有评估集
有 holdout
有人工复核与生产监控

AI 降本不是把最贵的模型换成最便宜的模型。

它是在质量不退的前提下，让更少的调用，用更少的 token，去调用最合适的模型。

没有 Evals，降本是赌博
有 Evals，降本才是工程

先看见成本，再守住质量，最后持续优化