

LambdaAgent PaaS 项目介绍

文档版本： 1.0 日期： 2026-05-28

一、整体定位

LambdaAgent PaaS 是一个基于 **λ -演算理论** 的 Agent 平台即服务（Platform-as-a-Service）框架。它把 AI Agent 的构建、运行、组合纳入到一套**可证明、可优化、可计费**的形式化代数语义之下，并通过 Web UI 让非技术用户也能直接使用。

整个项目分为三层：

Web UI	— 面向领域专家的可视化工作台	
PaaS	— Agent 平台 / 实例 / 计费	
LambdaAgent	— λ -演算驱动的 Agent 引擎	

一句话定位：让 **Agent** 像 **λ 表达式** 一样被构造、组合与计费——并把这一切藏到一个上手即用的浏览器界面背后。

二、LambdaAgent —— 核心引擎层

2.1 一句话定义

Agent $\equiv \lambda$ -term：每一个 Agent 都是一个带 **效应（effect）** 和 **成本（cost）** 标注的 λ 表达式，由 CEK 抽象机求值，遵循可证明的代数律。

LambdaAgent 不是又一个 Agent 框架，而是一套带语义的 Agent **计算模型**。

2.2 三大支柱

① CEK 求值器（Control-Environment-Kontinuation）

经典 λ -演算抽象机被改造成 Agent 调度内核： - **Control**：当前正在求值的 Agent 表达式 - **Environment**：变量绑定 + 工具 / 记忆 / 上下文 - **Kontinuation**：续延栈，天然支持 ReAct、Handoff、回溯

② Graded Effects（分级效应系统）

每个 Agent 项都有可静态推断的效应签名：

效应	含义
Pure	纯函数式调用，无副作用
LLM	调用大模型
Tool	外部工具 / IO
GroupChat / AsyncPar	并发协作
Handoff / Send / Receive	Agent 间通信

`infer_effect_for_term()` 在运行前就能告诉你这个 Agent 会"碰到什么"。

③ Cost Vectors（成本向量）

成本不再只是 token 计数，而是向量： `{tokens, latency, $, tool_calls, ...}` - 预测：求值前估算 - 实际：运行时累加 - 校验： `validate_cost()` 交叉比对，偏差即异常

2.3 代数律（Algebraic Laws）

LambdaAgent 强制 Agent 组合满足等式律：

<code>seq(seq(a, b), c)</code>	<code>≡</code>	<code>seq(a, seq(b, c))</code>	-- 结合律
<code>par(a, b)</code>	<code>≡</code>	<code>par(b, a)</code>	-- 交换律
<code>handoff(a, id)</code>	<code>≡</code>	<code>a</code>	-- 单位元

这意味着 **Agent** 编排可以被编译器 / 优化器重写，而不只是被解释执行。

2.4 与传统 Agent 框架的差别

维度	传统框架（LangChain / AutoGen 等）	LambdaAgent
抽象	类 + 回调	λ -term + 效应
组合	编排 DSL / 图	函数组合（可证等价）
成本	事后统计	向量 + 预测 + 校验
并发	ad-hoc	AsyncPar / GroupChat 一等公民
复用	复制代码改参数	Template + Instance 注入



三、LambdaAgent PaaS —— 平台层

PaaS 层在引擎之上提供工程化能力：实例管理、运行隔离、协议接入、领域部署。

3.1 核心机制

- **Agent = Template + Instance**：一个模板可派生 N 个领域实例，通过 `load_instance()` / `create_instance()` 注入领域数据。模板与数据彻底解耦，一次编写、多域复用。
- **Run Workspace**：每次运行隔离在 `{agent_dir}/workspace/run_{时间戳}/`，input / output / trace / config / cost 全留痕，永不删除。
- **成本预测与校验**：闭环可观测，预测与实际偏差即时报警。
- **多 LLM 接入**：已打通 小艺 **OpenClaw**（A2A WebSocket）、**agent67**（ReAct + done 工具）等协议。

3.2 QA Demo 四模式

右侧面板支持 **Wiki / BM25 / DirectLLM / RAG** 四种检索回答模式对比，已部署三个领域实例：

实例	部署地址	规模
maritime-qa	(已部署)	海事领域
finance-qa	(已部署)	金融领域
clean-qa	1.95.125.250:8083	49 文档 / 1527 chunks

3.3 项目状态

- **Phase 1 (P0) + Phase 2 (P1)** 已完成，共 175 个测试通过

- 三篇底层论文 (Paper I / II / III) 覆盖度约 **95% / 80% / 70%**
- 已修复关键设计缺陷：
 - DESIGN-01: `PassthroughHandler` 基类
 - DESIGN-07: 效应推断补全 (GroupChat / AsyncPar / Handoff / Send / Receive)
 - DESIGN-08: 成本预测 vs 实际交叉校验
- 协议: BSL 1.1 (≤ 10 用户生产环境免费, 2031-04-05 自动转 Apache 2.0)

四、Web UI —— 应用层

4.1 定位

CLI 给开发者, **Web UI** 给领域专家。一台本机、一个浏览器标签页, 就是完整的 Agent 工作台。

Web UI 把原本只能通过 CLI、YAML、`.env` 来驱动的平台, 包装成一个**零终端、零配置文件**的本地可视化应用。

4.2 技术栈

- 前端: React 18 + TypeScript + Vite + TailwindCSS
- 路由 / 状态: React Router 6 + Zustand + TanStack Query
- 后端: 直连现有 `agentpaas` REST API (不重新造后端)
- 部署形态: 本地单机 / 内网, 不依赖云服务

4.3 核心页面

模块	作用
SetupWizard	三步安装向导：环境检测 → 选 LLM 提供商 → 启动服务
Dashboard	系统状态、最近运行、Token 消耗一屏总览
Agents	智能体模板列表（创建 / 编辑 / 导入 / 删除）
AgentCreate / Edit / Import	表单化配置，告别手写 YAML
Chat	选择 Agent 直接对话，看到推理轨迹与成本
Knowledge / KnowledgeDetail	知识体管理：上传文档、构建索引、按域隔离
Providers	多 LLM 提供商管理

4.4 三大设计原则

① 零门槛安装

- 自动检测 Python / Docker 环境，缺什么提示什么
- 支持 **Python 模式** 与 **Docker 模式** 二选一
- 区分 **需要 API Key**（Anthropic / DashScope / DeepSeek / OpenAI / 智谱 / Moonshot...）与 **无需 API Key**（Claude Code Max Plan / Ollama 本地）

② 模板 / 实例双层管理

继承 LambdaAgent 的"一个模板，N 个领域实例"架构——UI 上一个 Agent 卡片可以展开多个领域实例（maritime / finance / clean...），互不污染。

③ 透明可观测

每次对话都直连后端的 Run Workspace，UI 可见： - 推理轨迹（trace） - 工具调用链 - Cost Vector 实时累加 - 历史回放

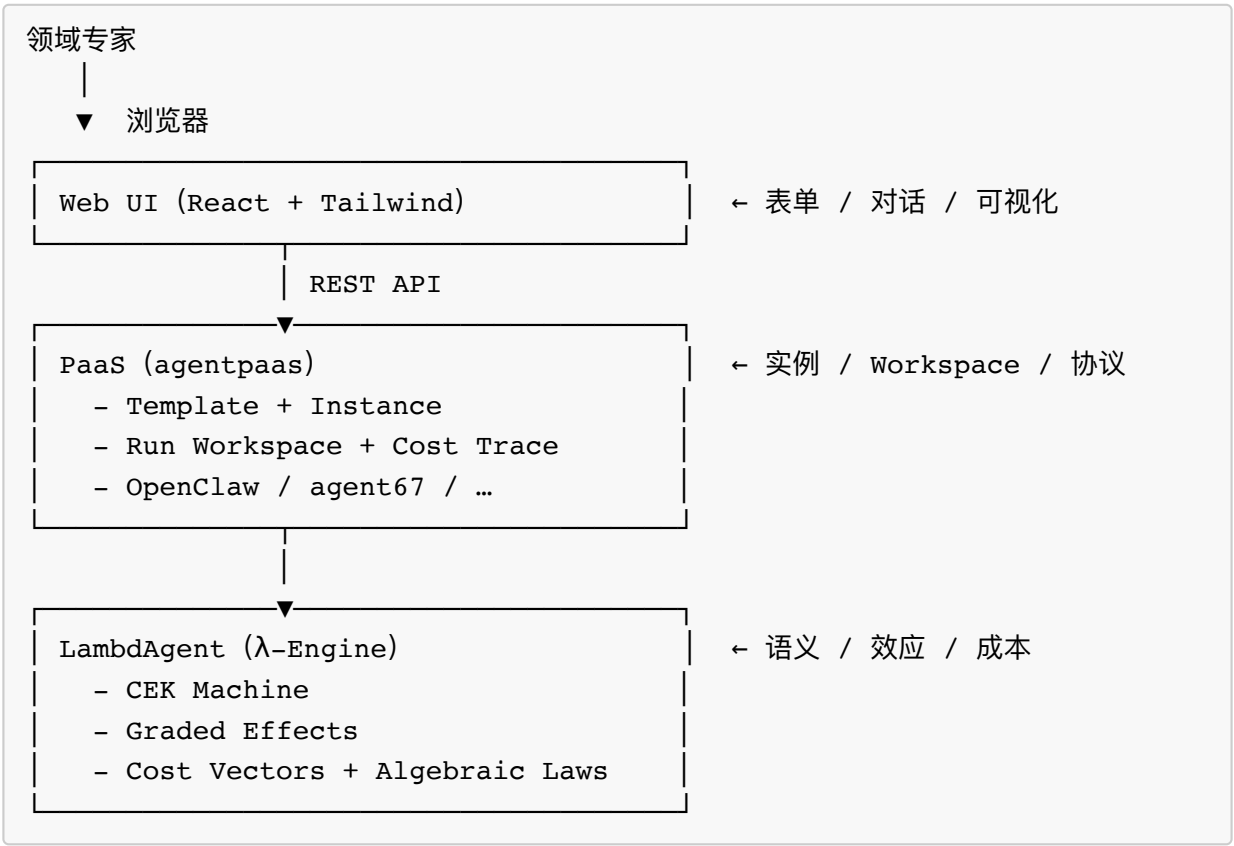
4.5 目标用户

用户	用 Web UI 做什么
领域专家（医、法、金融、海事...）	上传专业文档，三分钟造一个垂域问答 Agent
企业内部用户	开箱即用的对话端，不接触底层
科研人员	可视化配置 + 看到完整执行轨迹
学生 / 课堂	向导带新手装环境、跑 Demo

4.6 当前进展

- 首期需求文档 v1.0 ([WEBUIREQUIREMENTS.md](#)) 已交付
- 核心页面（10 个）已搭建完毕
- 已接通：Setup Wizard / Workspace 路由修复 / Agent 实例机制
- 最新迭代：补齐 **4.12 知识体管理需求**

五、总览：三层协作



六、一句话总结

LambdaAgent 让 Agent 第一次拥有了"可计算的语义"; **PaaS** 让这套语义能在多领域、多协议、多 LLM 之间复用; **Web UI** 让不会写代码的人, 也能用上这一切。

——这就是 **LambdaAgent PaaS**。