One Person Lab

OPL Cloud 白皮书

让复杂知识工作跨越本机、私有数据、远端资源与对外服务，仍然保持连续、受控、可复查

OPL Cloud 让工作从本机自然延伸到在线 Workspace、私有数据、远端资源、协作和 Agent 服务，同时让用户持续围绕成果工作，让权限、成本、证据与专业判断各归其位。

让复杂知识工作跨越本机、私有数据、远端资源与对外服务，仍然保持连续、受控、可复查

2026-07-15

Public whitepaper

目录

1 OPL Cloud 白皮书

让复杂知识工作跨越本机、私有数据、远端资源与对外服务，仍然保持连续、受控、可复查

发布日期：2026-07-15 最近修订：2026-08-16

适用对象：正在使用 AI 推进科研、基金、汇报、书籍或其他长期知识项目，希望沿用现有工作方式使用在线工作台、协作资源与远端计算，或者把成熟 Agent 能力提供给外部用户的个人、实验室和机构。

核心判断：OPL Cloud 让工作从本机自然延伸到在线 Workspace、私有数据、远端资源、协作和 Agent 服务，同时让用户持续围绕成果工作，让权限、成本、证据与专业判断各归其位。

2 定位摘要

OPL Cloud 是 One Person Lab 面向复杂知识工作的云端产品。用户继续围绕问题、材料、阶段、产物和审阅推进工作；系统在背后连接 AI、在线工作空间、账号策略、数据与计算资源、Agent 服务和可复查证据。

OPL Cloud 从复杂知识工作的连续性出发组织云端能力。它让长期工作在环境变化后仍保持上下文连续，让资源使用受到清楚约束，让每份结果都能回到它的来源与责任方。

OPL Cloud 以一个连续的产品体验承接本机工作、在线协作、远端资源与对外服务。用户不需要追随系统内部的变化，只需知道工作可以从原处继续，数据与权限仍受约束，结果仍能回到同一项目并接受复查。

OPL Cloud 希望兑现五个用户结果：

- 从本机开始，需要时再接入在线工作台或远端资源，原项目可以继续。
- 私有数据可以留在机构边界内，计算尽量去到数据所在的位置。
- 账号或协作成员只在关键动作前完成授权，日常工作保持连续。
- Agent 开发者可以把经过验证的能力提供为 API、嵌入组件或托管界面。
- 结果回到原工作链，并带有足够的来源、环境、审阅和继续线索。

3 为什么复杂知识工作需要不同的云

一次普通云任务通常可以用“上传、运行、下载”描述。长期知识工作还包含持续修改、授权、审阅、交接和专业判断。

研究者可能先在本机整理问题和材料，随后发现分析需要医院内网数据、实验室高性能计算集群和特定软件环境。协作成员需要知道谁可以访问数据、任务会消耗多少资源、输出应回到哪里；领域负责人还要判断结果是否足以支持论文中的结论。Agent 开发者还可能希望把成熟能力提供给外部 App 或网站。数周后，另一位成员需要理解这次分析从哪里来，并接着修改。

如果每遇到一个问题就切换到一套独立系统，用户会同时维护本地文件、云盘、作业队列、模型账号、审批消息、环境说明和人工日志。核心困难不是缺少工具，而是这些工具没有围绕同一个成果协作。

OPL Cloud 因此以五组用户矛盾为设计起点：

用户需要	常见代价	OPL Cloud 的设计选择
使用更强的 AI 与计算	被迫离开当前工作台	本机 App 与在线 Workspace 使用连续的工作模型
使用私有数据与机构资源	复制数据或暴露凭据	让执行靠近数据，并保留资源边界
治理权限与成本	每一步都经过管理后台	只在真正需要授权的边界介入
把 Agent 提供给外部用户	重复搭建 API、鉴权、运行环境和前端	用统一服务层承接发布与访问
长期解释和复查结果	依赖成员记忆与手工记录	让来源、环境、审阅和继续线索随结果返回

这五个选择共同指向一个原则：**让工作保持连续，让权力保持分离。**

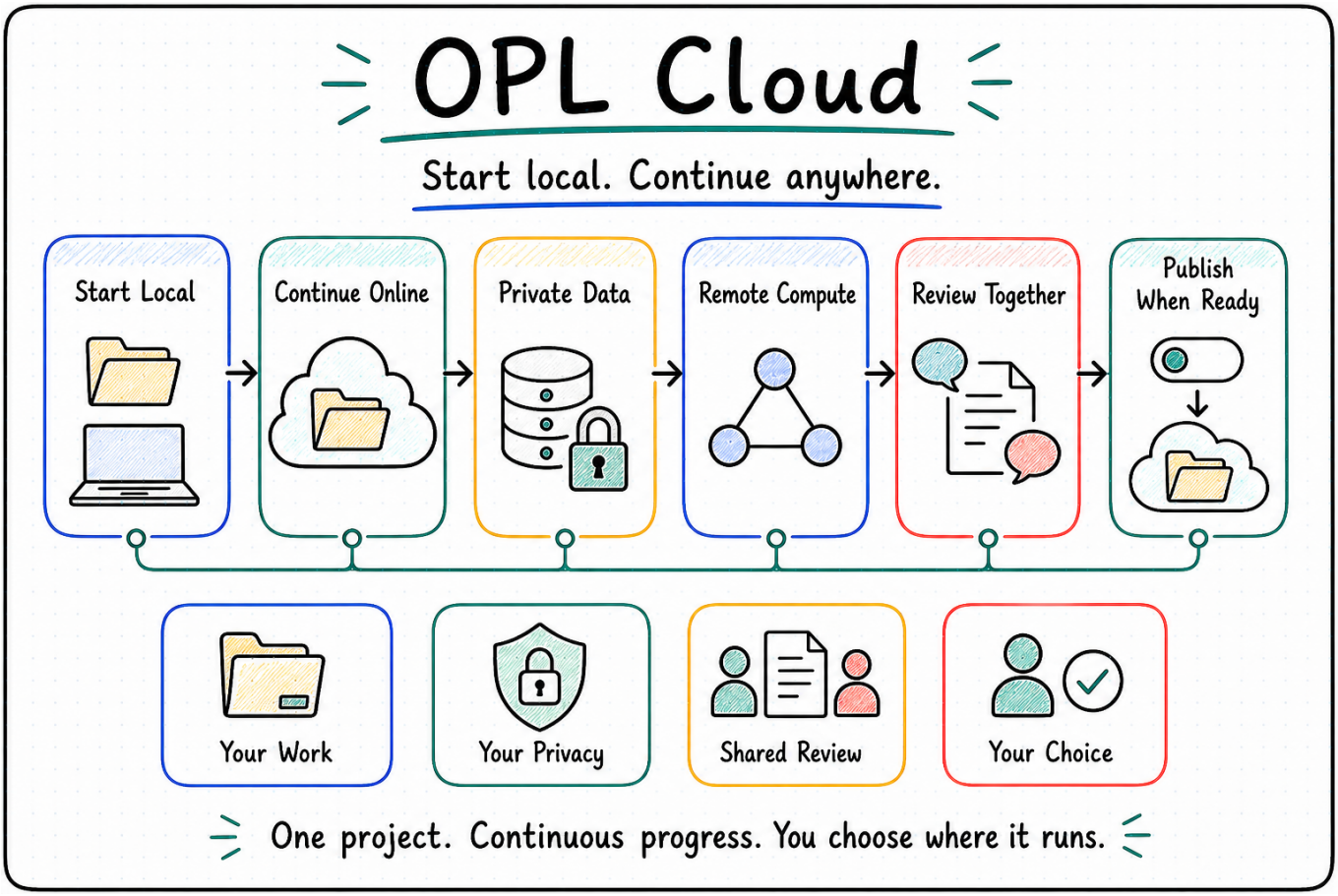

图 1: OPL Cloud 让同一项目从本机自然延伸到在线协作、私有数据、远端计算与服务交付。

4 OPL Cloud 的答案：连续工作面，分层责任

用户在 OPL App 与 OPL Workspace 中沿着同一套项目、材料、任务、产物、审阅和下一步继续工作。OPL Cloud 把在线工作空间、资源连接、协作与服务治理自然接入这条工作链。

从整个 OPL 生态看，OPL Base、OPL App、OPL Packages 和 OPL Cloud 共同组成四层产品体系。Base 提供共同运行底座，App 提供统一工作台产品，Packages 提供专业能力，Cloud 提供账号、协作、在线 Workspace、远端资源和治理。OPL App 是用户统一的产品工作入口，Cloud 则按照项目场景动态接入在线 Workspace、远端资源、协作与治理能力。能力可以持续演进，用户面对的项目、材料、动作、结果和 继续入口保持一致；账号、资源、费用与证据责任也不会因为产品演进而转移。

```
用户工作面          OPL App / OPL Workspace
                          |
AI 接入             OPL Gateway
Agent 服务           OPL Serve
资源连接与执行       OPL Fabric
账号政策与批准       OPL Console
来源与运行证据       OPL Ledger
专业质量判断         领域 Agent 与人类负责人
```

分层为每项责任指定清楚的位置：

- **Gateway** 让模型接入、路由和用量有稳定入口。
- **Serve** 让成熟 Agent 能力以统一方式面向外部用户，并承接服务访问与治理。
- **Fabric** 连接计算、存储、环境、数据源与外部系统，把一次任务绑定到获准资源。
- **Console** 服务用户与管理员，负责账号、Workspace、协作角色、额度、策略、批准和费用视图。
- **Ledger** 连接计划、批准、输入、环境、输出、审阅与继续线索。
- **领域 Agent 与人类负责人** 保留专业策略、证据判断、质量结论与最终决定。

因此，一个用户可以只理解“我要完成什么、系统准备使用什么、结果在哪里”。产品内部的责任边界，为这种简单体验提供可靠基础。

5 稳定能力语言与连续产品体验

OPL 使用一组跨产品的能力域来保持长期一致的语言：工作空间、资源连接、账号控制、运行证据、AI 接入和智能体服务等名称，描述的是稳定的责任和用户问题。同一个能力域可以由不同产品共同完成，最终判断始终由明确的产品、服务或领域负责人给出。

用户始终通过同一个 OPL App 产品体验理解项目和推动工作。Cloud 根据任务需要动态装配 账号治理、在线 Workspace、资源连接、运行证据、AI 接入和 Agent 服务；不需要的能力 不会打断当前工作，需要新增能力时也不要求用户重新组织项目。

这种设计让产品可以持续演进，同时保护体验连续性。能力升级或组合变化之后，项目身份、材料上下文、权限预期、结果证据和下一步仍然连得起来。用户感受到的是能力自然扩展，而不是在不同产品载体、版本节奏或内部组件之间迁移。

6 OPL Cloud 的五大设计原则

6.1 一、成果优先于基础设施

用户首先看到问题、材料、阶段、产物和下一步。容器、队列、存储或内部服务只在资源选择影响权限、成本或复查时进入界面。

这使复杂计算仍然表现为工作的一部分：用户批准“为这项分析使用实验室集群和指定环境”，随后收到图表、报告和审阅结果；作业编号和运行日志留在诊断层。

6.2 二、本机与云端是同一条工作链

本机 OPL App 适合个人控制、敏感材料和日常编辑；在线 OPL Workspace 适合远程访问、协作和托管执行。两者共享同一套产品语言，分别承载同一工作模型在本机与云端的工作面。

用户可以从本机开始，在需要在线访问或更强资源时选择合适的 Workspace，再把结果带回原项目。环境改变后，任务身份、材料引用、产物关系和继续入口仍然保留。Workspace 与对外 Agent Service 是不同对象：前者承载用户自己的工作，后者把成熟能力提供给外部消费者。

6.3 三、先形成计划，再授予资源权力

正式远端任务先形成一份可理解的资源计划：要访问哪些输入、使用什么环境与计算、写入什么位置、预计消耗什么、如何停止。个人、实验室或机构只对看得懂的计划授权。

这把危险动作和普通工作分开。低风险读取可以按既定策略继续；敏感数据出域、高成本计算、共享连接器、服务公开和资源变更则在明确边界上停下来。

6.4 四、管理面与工作面各司其职

Console 服务账号所有者与管理员：协作角色、权限、预算、额度、策略和批准。研究者、作者和设计者仍在 App 或 Workspace 中完成工作。

这种分工避免管理后台吞噬专业工作，也避免专业工作绕过资源与账号治理。用户在一个地方创作和审阅，在必要时理解并批准资源与成本；系统在背后保持两类责任的一致连接。

6.5 五、证据连接全程，专业结论归专业责任方

Ledger 让计划、批准、环境、输入、输出和审阅彼此可追溯。记录负责说明来路，科研结论由医学研究 Agent 和研究负责人判断；基金、视觉交付和书稿也各有自己的质量边界。

OPL Cloud 负责让判断有来路、结果能复查、工作可接力。运行成功只能说明任务完成了某一步，专业上是否成立仍由真正理解该领域的人和 Agent 决定。

7 能力版图：围绕用户问题，而不是系统目录

OPL Cloud 的能力按用户在一条工作链中遇到的问题组织。每个问题都有清楚的主要责任方。

用户问题	主要责任方	用户获得什么
我在哪里继续工作?	OPL App / Workspace	连续的项目、任务、产物与审阅体验
AI 从哪里来、用了多少?	OPL Gateway	稳定模型入口、路由与用量信号
怎样把 Agent 提供给外部用户?	OPL Serve	API、嵌入组件、托管界面与服务治理
数据、工具和计算怎样接入?	OPL Fabric	有边界的资源连接、执行与结果回收
账号允许谁使用或发布什么?	OPL Console	账号策略、批准、额度与管理视图
这次运行发生了什么?	OPL Ledger	来源、环境、审阅与继续线索
结果在专业上是否成立?	领域 Agent / 人类负责人	领域质量判断、修订意见与交付决定

界面连接各自可靠的事实来源，为用户形成一致体验；资源、服务、证据和专业判断则由最合适的责任方维护。

8 从成熟 Agent 到对外服务

成熟 Agent 不只可以在个人工作台中使用，也可以成为其他产品或团队可以调用的服务。OPL Cloud 把“开发能力”“提供服务”和“一次具体调用”分开，让每一层都有清楚责任。

经过验证的 `Agent` 能力
-> 形成可发布版本
-> 配置服务对象与访问政策
-> `API / Embed / Hosted UI`
-> 一次具体调用或持续会话
-> 结果、用量与证据返回

OPL Serve 提供三种交付方式，并让它们共享一套服务原则：

方式	面向谁	用户价值
API	已有 App、网站或后端	用稳定接口接入成熟 Agent 能力
Embed	已有网站中的交互区域	在原产品中加入受控的 Agent 体验
Hosted UI	需要现成前端的发布者	用任务、报告、流程或对话模板快速提供服务

三种方式共享身份、访问策略、额度、运行反馈和结果证据。发布者仍然负责 Agent 的专业承诺、内容和客户关系；OPL Cloud 负责让服务能够被安全、稳定、可解释地访问。

9 一条典型的专业工作链

下面的实验室场景，比模块目录更能说明 OPL Cloud 为什么这样设计。

9.1 1. 从本机形成问题

研究者在 OPL App 中整理研究问题、分析计划和已有材料。患者级数据仍留在医院或机构存储，本机项目只保留允许使用的引用和上下文。

9.2 2. 选择合适的专业能力

项目选择医学研究 Agent 与所需分析能力。用户看到这些能力能做什么、适用于什么任务，以及当前是否可以使用，不需要理解其安装和运行细节。

9.3 3. 形成资源计划

任务发现本机算力不足，于是形成一份可读计划：在实验室集群上运行、访问某个私有数据引用、使用指定分析环境、把脱敏结果写回项目输出区，并说明预算、停止方式和预期产物。

9.4 4. 在关键边界批准

Console 根据账号与机构策略检查成员权限、资源额度、数据边界和预算。策略已覆盖的动作可以继续；涉及敏感数据或高成本资源时，由有权的人确认。

9.5 5. 在数据所在处执行

Fabric 连接获准资源，提交并监控任务。数据默认留在资源所在位置；凭据和原始数据继续由资源提供方管理。

9.6 6. 把结果带回工作台

计算完成后，图表、汇总和运行线索回到原项目。用户仍在熟悉的 App 或 Workspace 中查看结果、比较版本和决定下一步，基础设施页面只在诊断与管理时出现。

9.7 7. 交给领域独立审阅

医学研究 Agent 对方法、数字、图表与主张的对应关系进行专业审阅。运行完成不等于研究结论成立；发现问题时，修订意见回到同一工作链，必要时再次计划和执行。

9.8 8. 留下可继续的证据

系统连接资源计划、批准、环境、输入、输出、审阅结果、负责人和继续入口。几个月后，协作成员仍能理解当时发生了什么，并沿完整上下文继续工作。

10 四类用户路径

同一设计可以随着工作场景逐步展开，让不同用户从最适合自己的工作入口开始，并在需要跨环境协作时自然接入 Cloud。

个人用户 可以始终以本机 App 为主，只在需要在线访问、AI 或远端计算时接入 Cloud。私有项目保持原有材料位置。

Agent 开发者 可以把成熟能力提供为 API、嵌入组件或托管界面。开发者仍对 Agent 的专业承诺、内容和终端客户关系负责。

实验室与协作团队 可以共享批准过的模型、能力、连接器、环境和计算资源。成员在各自工作面推进项目，管理员只处理账号或机构策略，领域负责人继续对专业结果负责。

企业与机构 可以让内部数据库、私有存储、工具 API 和计算资源保持原有管理边界，同时把它们接入标准工作链，统一管理权限、预算和可复查证据。

四条路径共享同一组设计判断：从成果出发、随工作场景扩展、关键动作授权、责任各归其位。

11 六类信任机制

11.1 范围可信

用户知道材料在哪里、任务准备访问什么、是否发生外部传输。敏感数据默认留在用户工作空间、机构存储或私有环境，Cloud 只连接完成工作所必需的信息。

11.2 执行可信

远端任务说明计算位置、软件环境、输入、输出和停止方式。Fabric 的职责是忠实执行获准计划，并返回可检查结果。

11.3 版本可信

用户能知道正在使用或发布的是哪一版能力，变化有清楚来源。工作台、服务与资源执行都引用同一份经过确认的版本，不各自维护互相冲突的副本。

11.4 服务可信

对外 Agent 服务有清楚身份、访问政策、运行状态和责任方。API、嵌入组件和托管界面遵循一致的访问、反馈与证据原则。

11.5 结果可信

运行证据交给领域审阅。最终质量、发布、提交或交付决定来自对应领域的专业 Agent 和必要的人类负责人。

11.6 接力可信

重要结果带有负责人、输入输出、审阅结果和继续入口。协作成员未来可以重新理解和修订，不依赖某个人记住全部过程。

12 本文边界

这份白皮书面向用户解释 OPL Cloud 的设计理念与目标产品边界，回答“为什么这样设计会更好用、更可信”。功能清单、安装说明、服务状态、价格、发布时间与正式公告由对应产品页面提供。

本文呈现 OPL Cloud 的产品选择与设计承诺。具体能力的当前可用状态由正式产品页面持续更新；读者可以据此判断 OPL Cloud 是否对复杂知识工作的真实矛盾做出了完整、专业和一致的回答。

13 结语

好的云端产品让用户继续思考问题、审阅结果和推进成果，同时在需要时获得更强 AI、在线工作空间、私有数据连接、远端计算、协作和对外 Agent 服务。

OPL Cloud 选择用连续工作面承接用户，用 Serve 提供成熟 Agent 能力，用 Fabric 连接资源，用 Console 克制地治理账号策略，用 Ledger 保存可追溯证据，再把专业判断交还领域 Agent 与人类负责人。

这套设计的价值来自清楚责任：工作可以跨环境继续，数据始终保持边界，管理服务创作，证据支持判断。复杂性被放在正确的位置，用户因此获得连续、可控、可复查的云端体验。