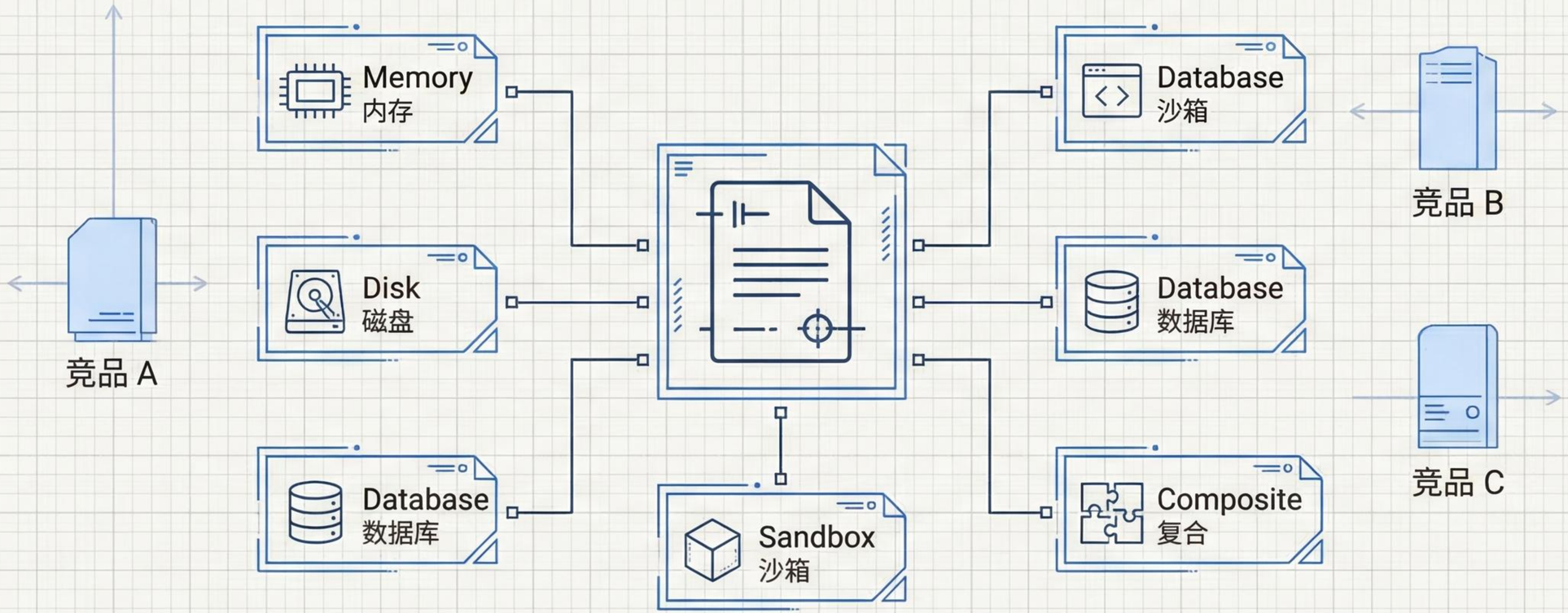
Deep Agents 实战

第 2 讲：Context Engineering 与竞品对比



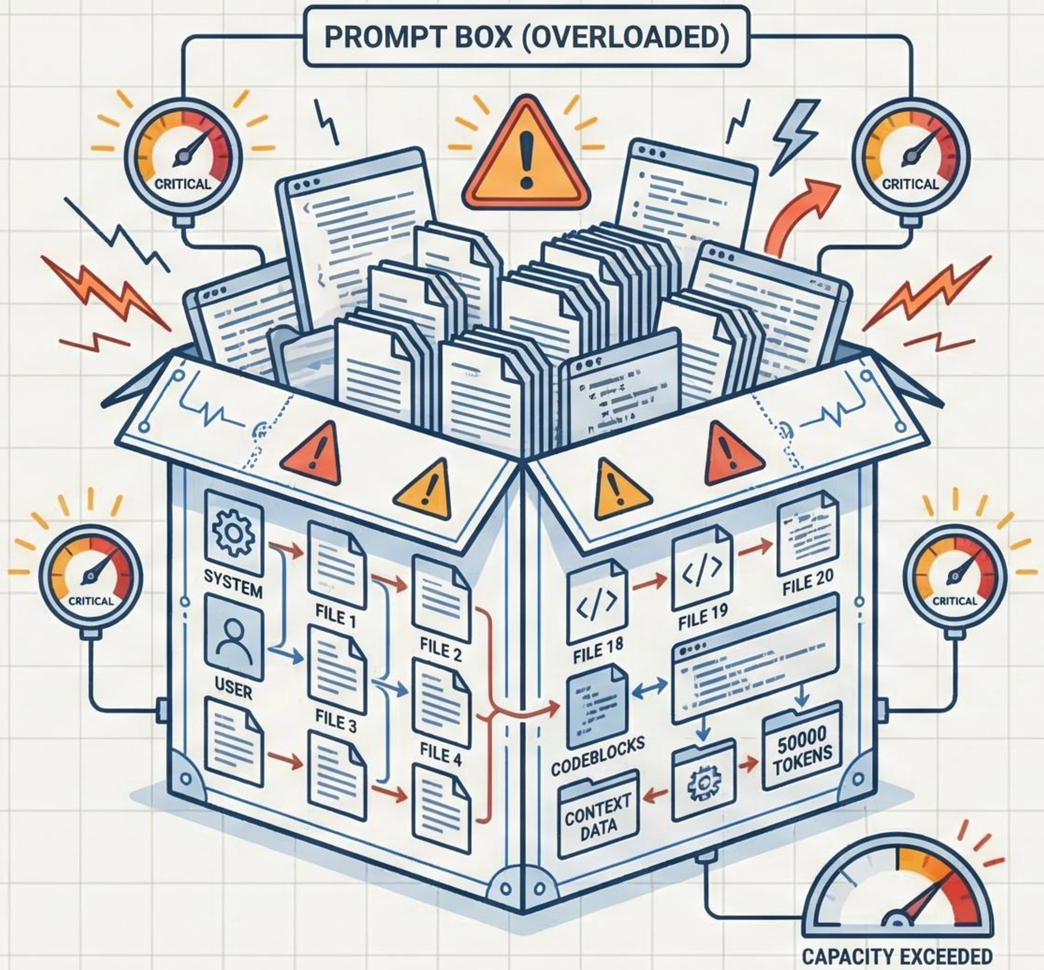
上一讲回顾：Agent 开发的三个层次

Runtime → Framework → Harness

今天的问题：Harness 的核心理念是什么？与竞品有何不同？

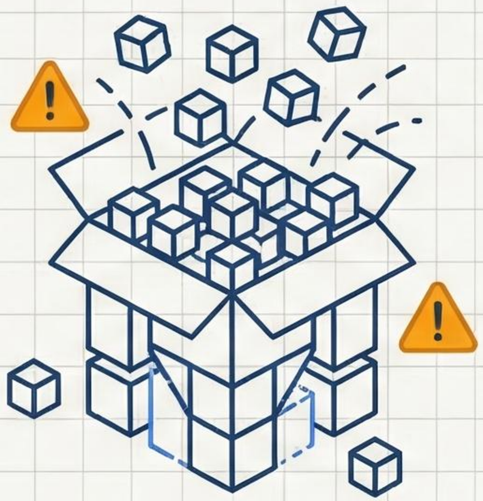
传统做法：Prompt Stuffing

把所有信息塞进提示词

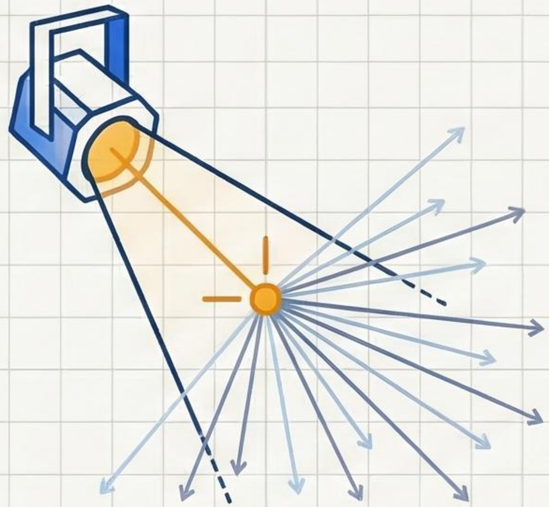


- System：你是一个编程助手
- User：请帮我重构 src/ 下的代码
- [附带：20 个文件的完整内容，共 50000 tokens]
- 结果：LLM 被迫一次性消化所有信息

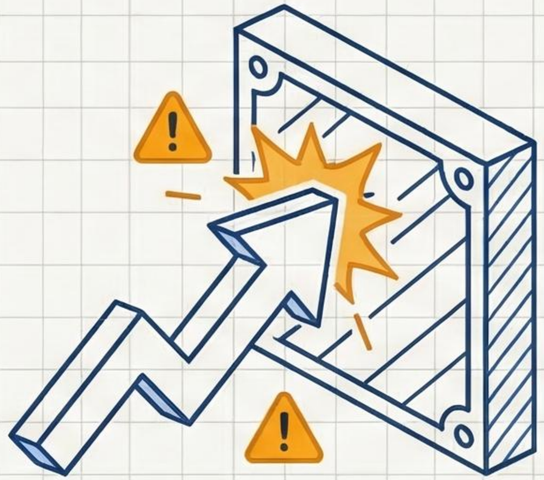
Prompt Stuffing 的三个致命问题



上下文窗口溢出：LLM 有 token 上限，文件一多就装不下



注意力稀释：信息越多，LLM 对关键信息的关注度越低



不可扩展：无法处理任意规模的项目

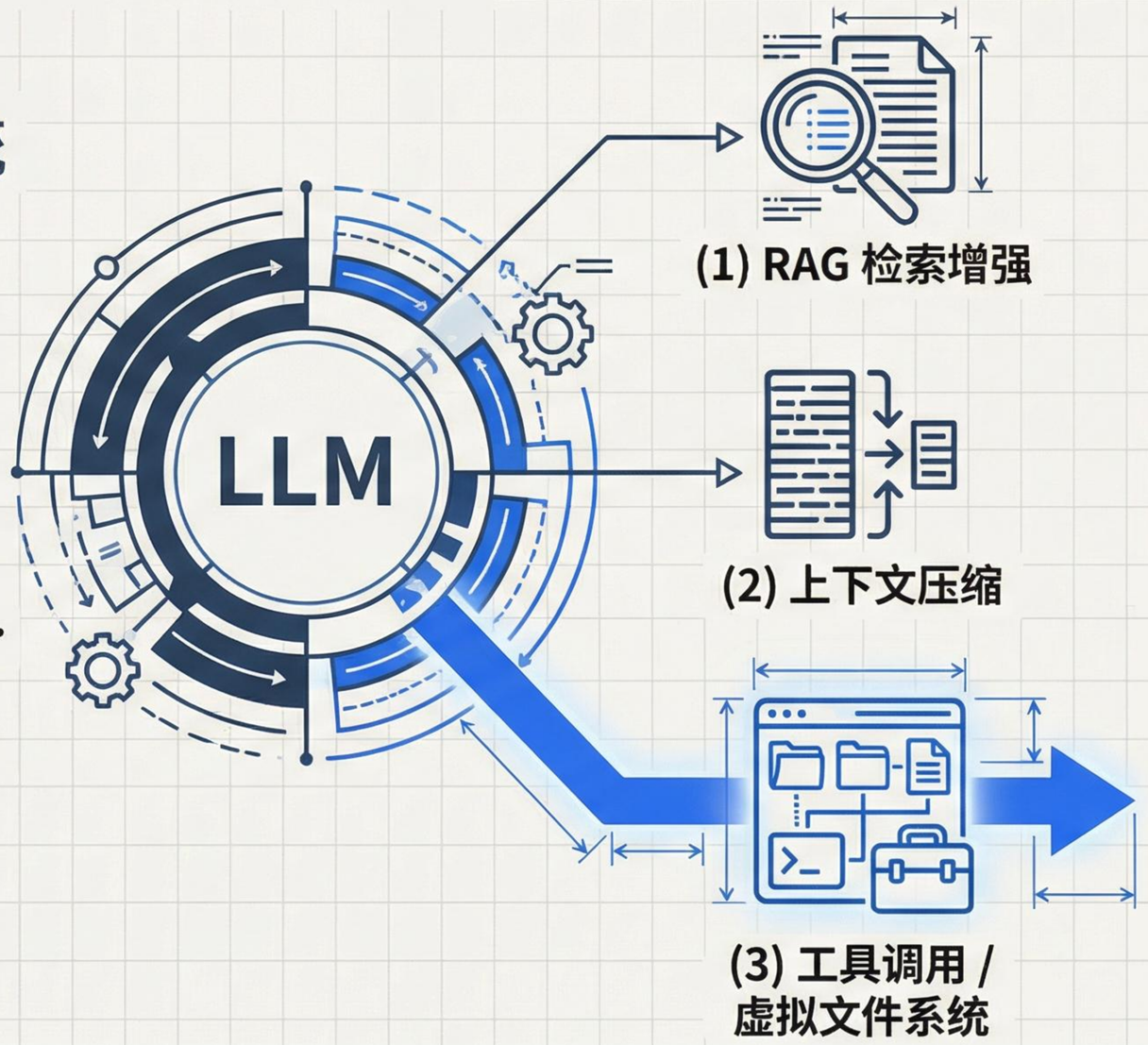
Context Engineering：上下文工程

围绕 LLM 构建高效获取信息的系统

- 核心思想：不是把信息塞给 LLM，
  而是让 LLM 能按需获取信息

- 不同产品有不同的实现策略
- RAG 检索增强 / 上下文压缩 / 工具调用 …

- Deep Agents 的选择 → **虚拟文件系统**

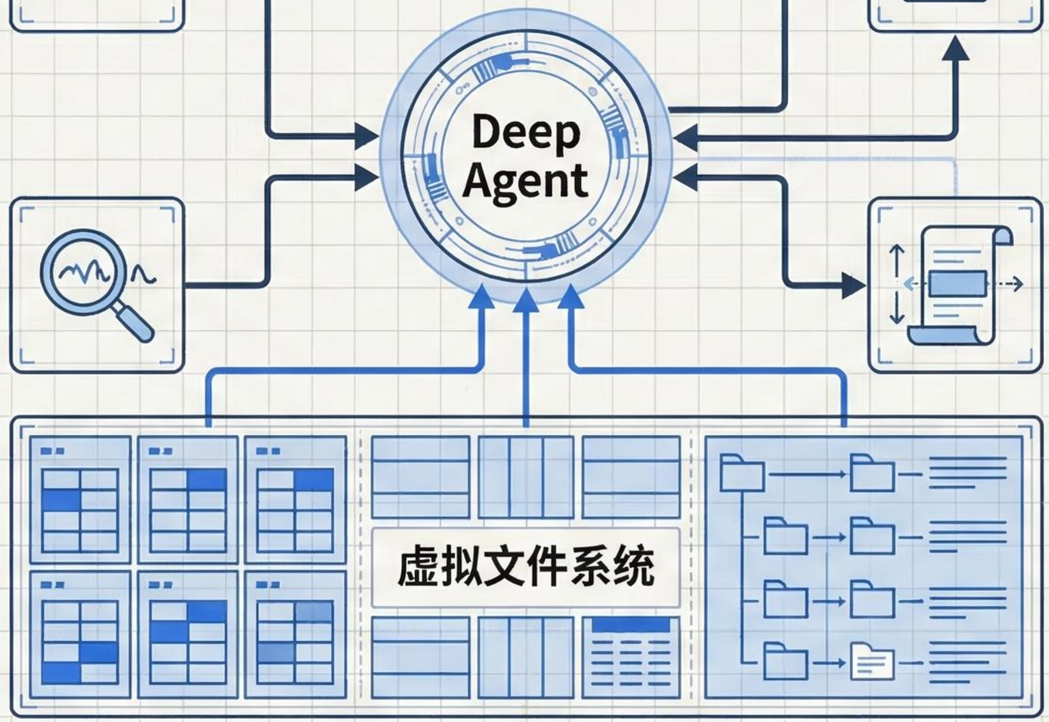


Deep Agents 的解法：虚拟文件系统

让 Agent 像人类一样按需获取信息

read_file
— 按需读取文件内容

write_file
— 记录中间结果



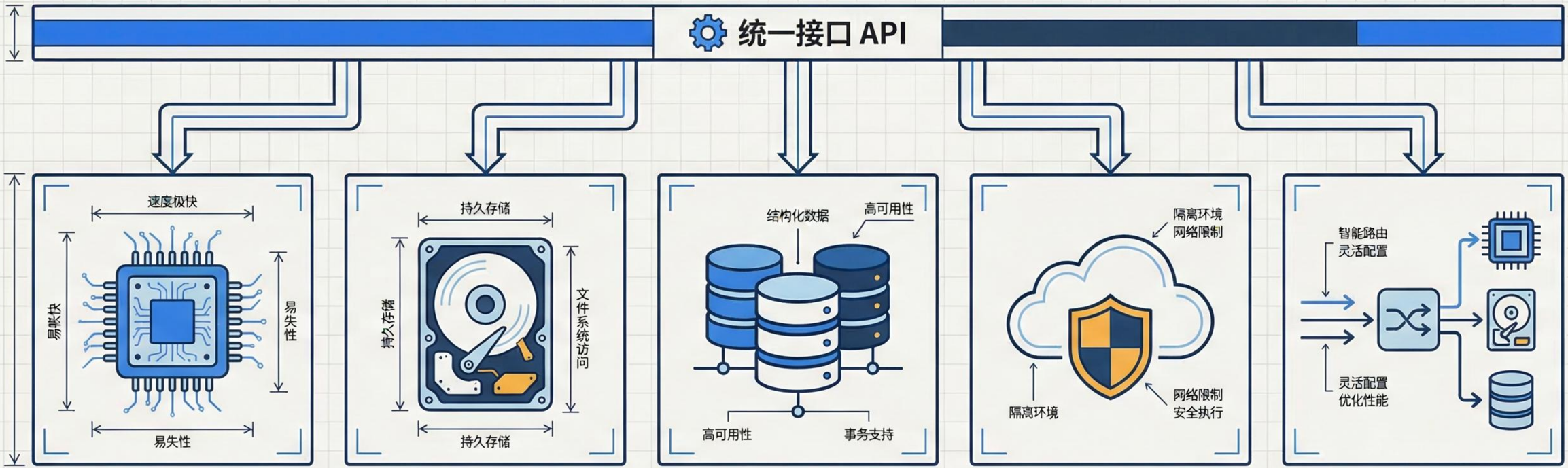
grep / glob
— 搜索定位信息

offset / limit
— 大文件只读取需要的部分

核心理念：上下文里只保留当前步骤真正需要的信息

可插拔的存储后端

统一接口，多种实现



内存存储 —
开发调试，临时数据

本地磁盘 —
处理真实文件

持久化数据库 —
跨会话保持记忆

远程沙箱 —
安全执行代码

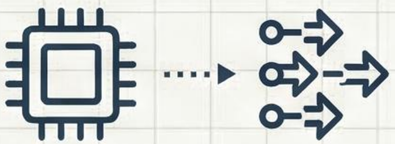
混合模式 —
不同路径路由到不同后端

三大 Agent Harness 的定位

Deep Agents vs Claude Agent SDK vs Codex SDK

Deep Agents

通用 Agent，模型无关
（100+ 模型）



100+ 模型支持

Python　TypeScript

Claude Agent SDK

自定义 AI 编程 Agent，
绑定 Claude 系列

绑定 Claude 系列

Custom SDK

Codex SDK

预构建的编程 Agent，
绑定 OpenAI 系列

绑定 OpenAI 系列

Pre-built SDK

核心能力对比：各家的独有优势

Deep Agents

Deep Agents 独有

- 模型灵活性
- 长期记忆
- 虚拟文件系统 + 可插拔后端
- Sandbox-as-Tool
- LangSmith 可观测性

Claude Agent SDK

Claude Agent SDK 独有

- Claude 深度集成
- Hooks 拦截系统
- 自定义 HTTP/WebSocket 层

Codex SDK

Codex SDK 独有

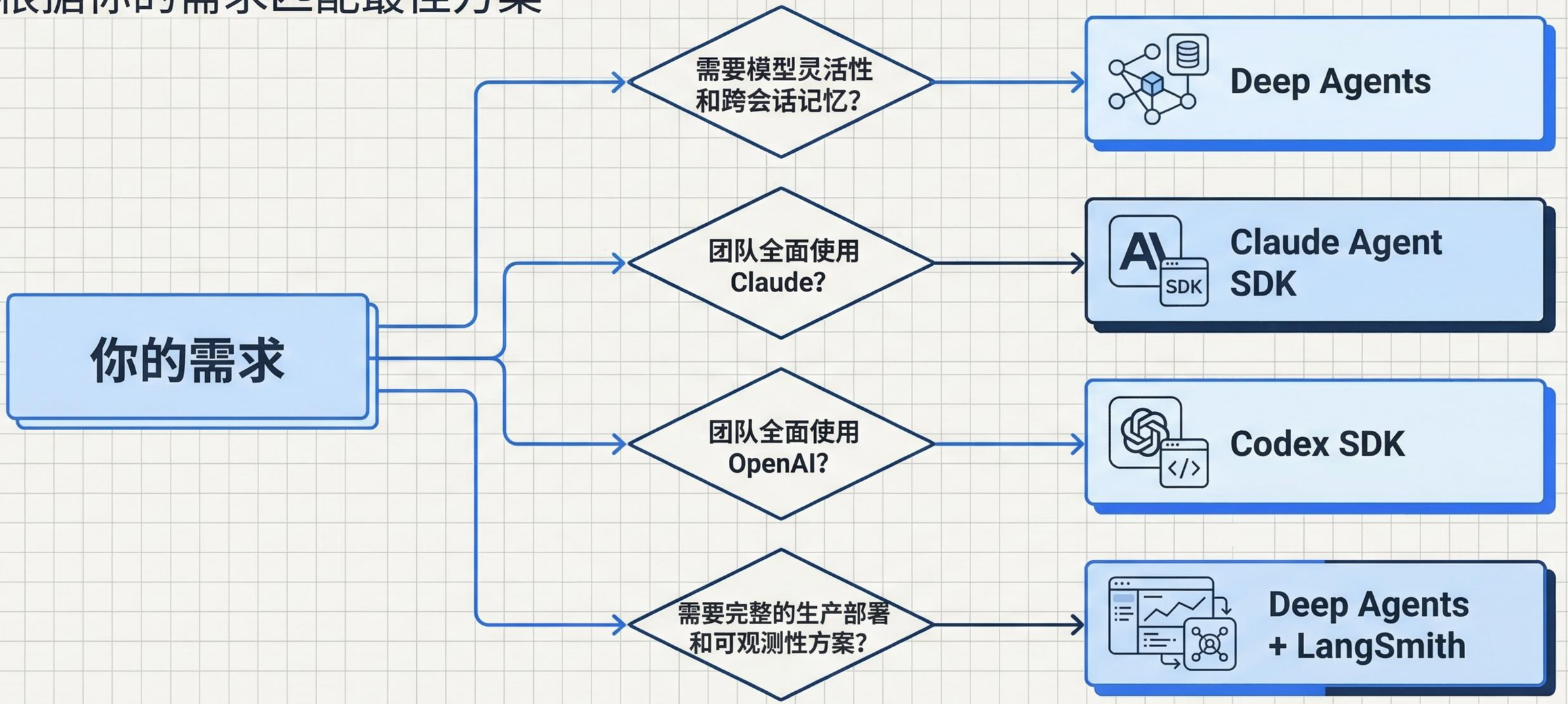
- OS 级沙箱模式
- 内置 MCP Server
- 云端执行环境

共同能力

Deep Agents	Claude Agent SDK	Codex SDK
文件读写	文件读写	文件读写
Shell 执行	Shell 执行	Shell 执行
搜索	搜索	搜索
规划	规划	规划
子 Agent	子 Agent	子 Agent
MCP	MCP	MCP
人机协作	人机协作	人机协作

如何选择?

根据你的需求匹配最佳方案



本讲回顾

- Context Engineering = 为 LLM 构建高效获取信息的基础设施
- 虚拟文件系统 + 可插拔后端 = Deep Agents 的核心架构
- 与 Claude Agent SDK / Codex SDK 对比：模型无关 + 长期记忆是最大差异
- 下一讲：5 分钟快速上手，构建你的第一个 Deep Agent