Deep Agents 实战

第 14 讲：沙箱执行 — 安全运行代码

一张图看懂代码执行

三层能力：Backend、dcode 与托管沙箱

同一目标：安全地产生可审查、可下载的结果

沙箱 Backend：多出的不只是一个工具

普通 Backend 解决文件读写；沙箱 Backend 额外提供隔离的 Shell 执行

隔离边界：保护什么，不保护什么

沙箱隔离宿主副作用，但不会自动让 Agent 或输入变得可信

沙箱有效保护

- 宿主文件、进程与环境变量
- 错误脚本、依赖安装与测试失败
- 执行环境的故障隔离

仍需额外治理

- 上下文注入：诱导执行危险命令
- 网络外传：HTTP / DNS 带走数据
- 不可信产物：代码、报告、构建物

不放秘密

限制网络

高风险 HITL

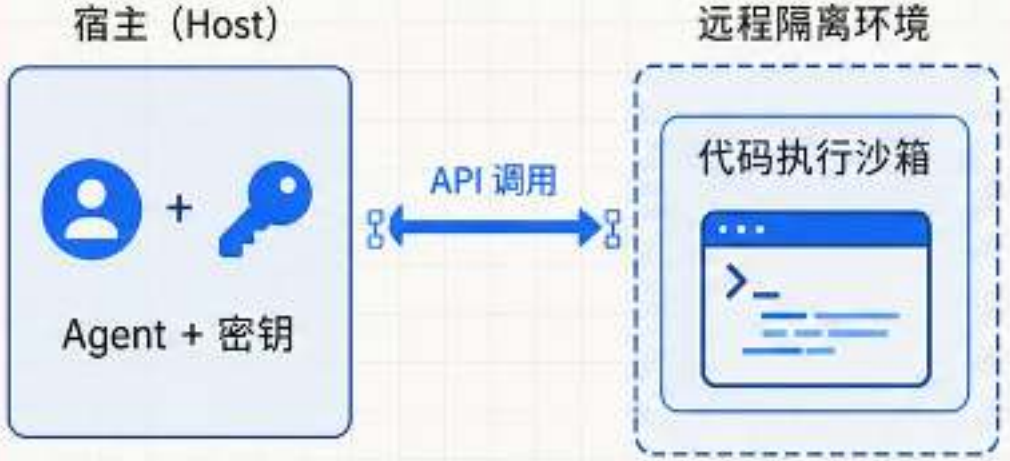
审查产物

监控出站

两种集成模式

默认推荐 Sandbox as tool：Agent 与密钥留在宿主，代码在远程隔离环境执行

Sandbox as tool 推荐



优势

- 密钥不入沙箱
- 故障不丢 Agent 状态
- 可并行多沙箱

VS

Agent in sandbox

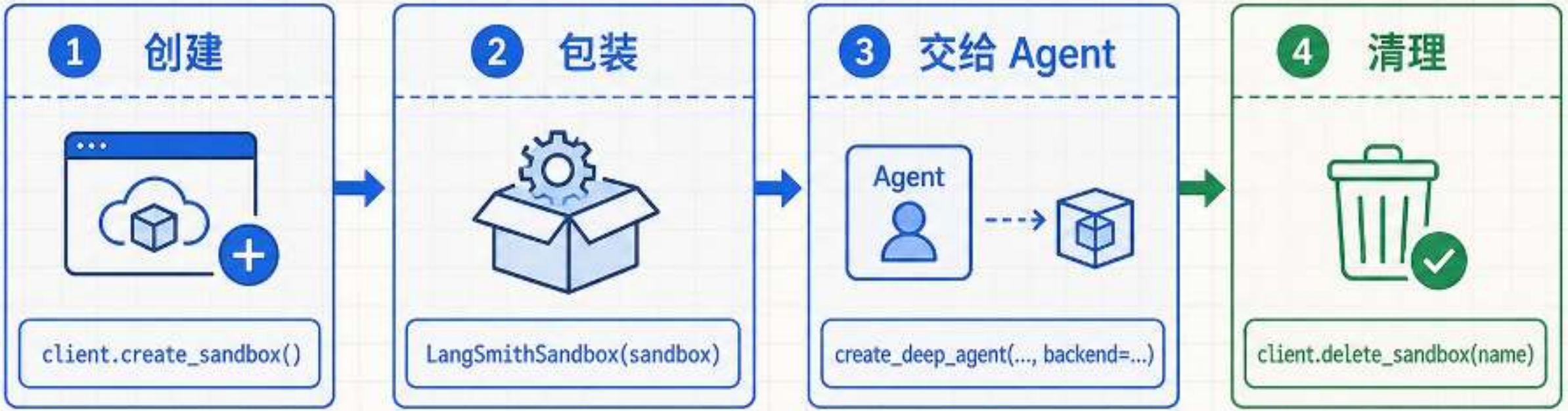
劣势

- 需重建镜像
- 额外通信层
- 凭证外传风险

仅在必须复现完整运行时、且 Provider 已妥善处理通信时再考虑

4 步接入远程沙箱

把 Provider 创建出的环境包装成 Backend，记得用 finally 或上下文管理器清理



运行前可用 backend.execute("python --version") 做健康检查；运行结束必须回收远程资源。

Provider 选择：先看语义，再看名称

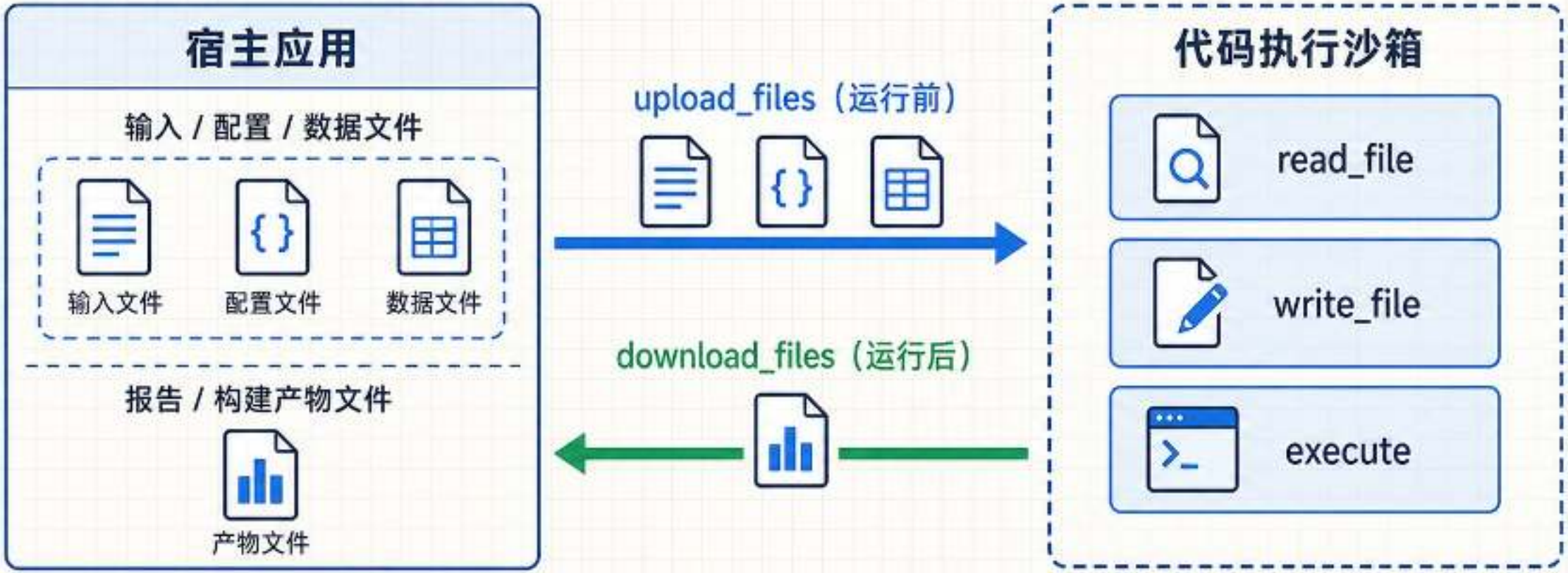
同一套工具语义，不代表相同的创建、清理、网络与凭证能力

运行区域	生命周期 / 回收	网络与凭证	镜像与依赖
LangSmith	AgentCore	Daytona	E2B
快照 · Auth Proxy	CodeInterpreter	云端开发环境	第三方 Backend
Modal	NVIDIA OpenShell	Runloop	Vercel
容器 · 网络控制	自动清理	Devbox	Sandbox.create()

上线前逐一核对 Region、TTL、网络策略、凭证控制与镜像能力

文件的两个平面

Agent 工具只操作沙箱内部；宿主应用通过 Provider 原生 API 跨越边界

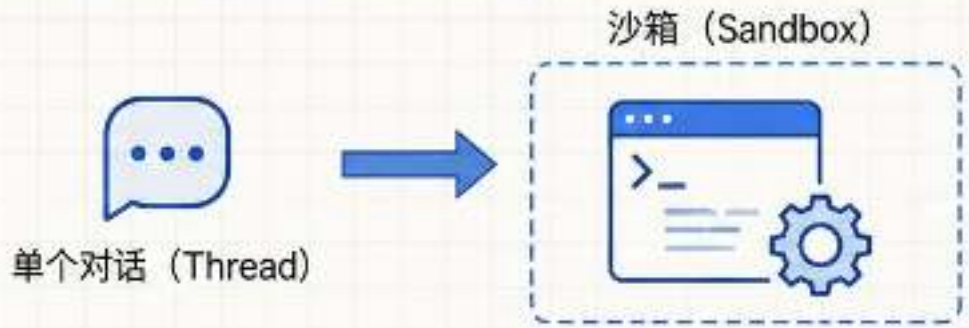


上传和下载是输入审查与产物审查的天然边界

生命周期与作用域

先决定环境属于一段对话，还是属于一个长期维护的 Assistant

Thread-scoped（默认）



- 一个 thread_id 对应一个沙箱
- 同线程复用；不同任务隔离
- idle_ttl_seconds 自动回收

`thread_id → sandbox`

Assistant-scoped

- 多个对话共享仓库、依赖与缓存
- 适合长期代码库或预装环境
- TTL、snapshot / reset、周期清理

`assistant_id → sandbox`

任务隔离 | 环境连续 | 状态累积风险 | 快照 / 重置

Deep Agents Code：本机循环，远程执行

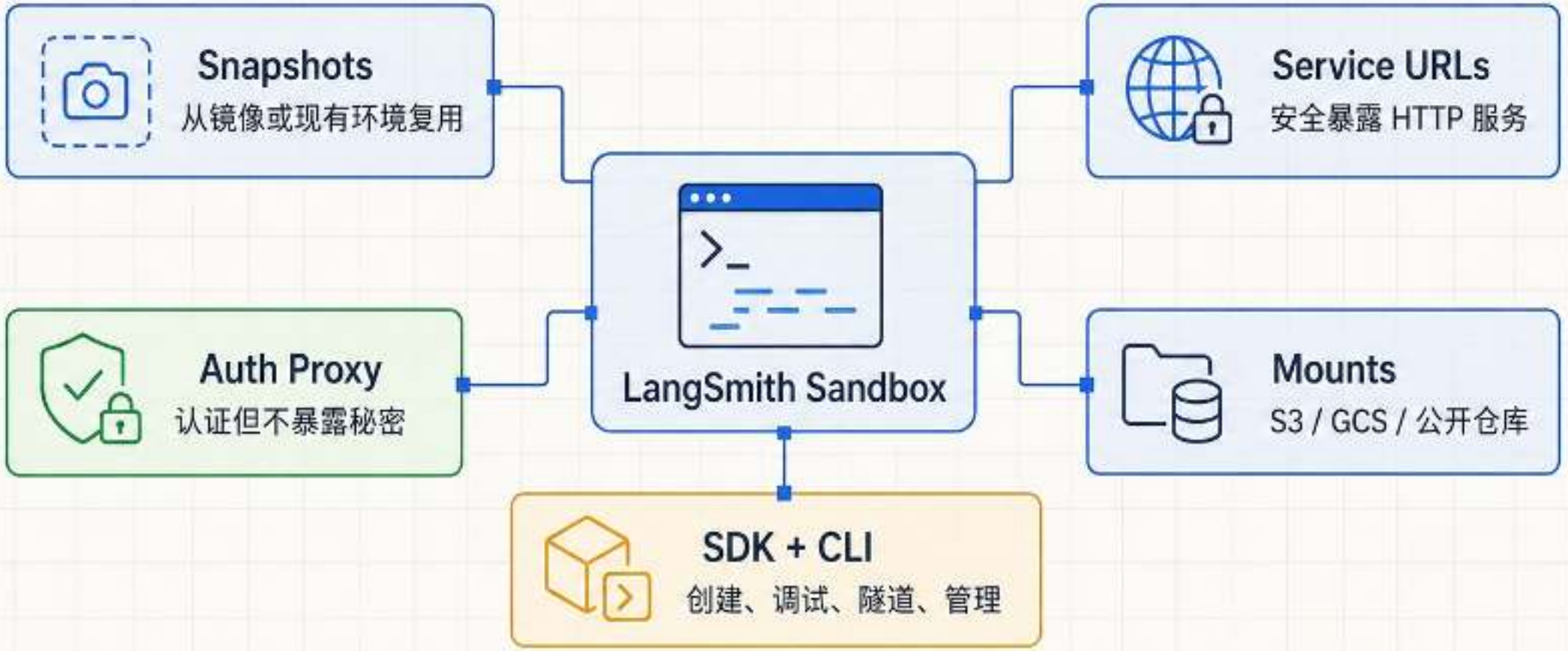
dcode 不在本地文件系统执行工具；它把文件操作与 execute 定向到 Provider 沙箱

配置声明 > 第三方 entry point > 内置 Provider

ⓘ 裸 --sandbox 必须放在命令末尾

LangSmith 托管沙箱：不止是一个 Backend

Sandbox 既可被 Deep Agents 包装，也可通过 SDK、CLI 和托管资源独立使用



Auth Proxy：沙箱请求可以被认证，但 Agent 看不到长期秘密

两个扩展点：Backend 与 dcode Provider

一个面向 Python Agent，一个面向命令行用户

Python Agent 集成

- 实现 SandboxBackendProtocol
- 重点实现 execute() 与结果映射
- 文件工具由沙箱基类构建

```
create_deep_agent(..., backend=AcmeSandbox(...))
```

Deep Agents Code 集成

- 实现 SandboxProvider
- 注册 entry point 或 config.toml
- 声明 working_dir、snapshot / ID 能力

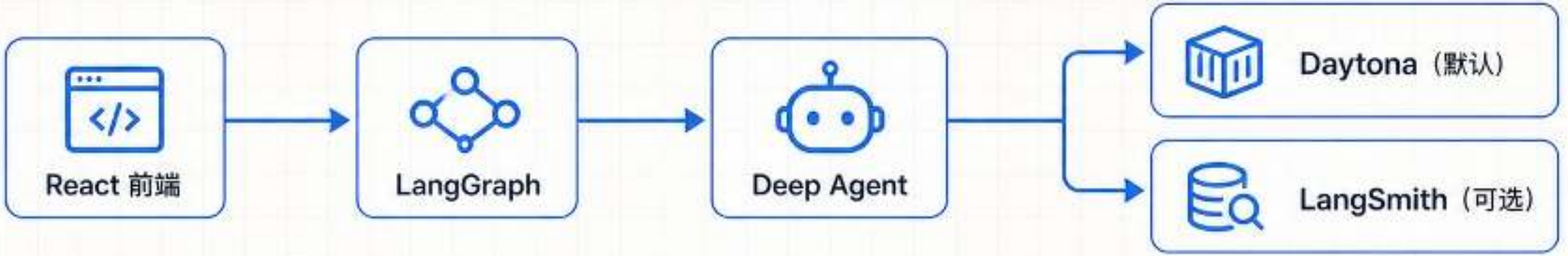
```
dcode --sandbox acme
```

不要把 class_path 指向不受信任模块：它会导入并执行本地 Python

AgentSeek 沙箱模板

一条命令跑起 Sandbox Coding Agent

```
>_ uv tool install agentseek-cli
>_  agentseek create deepagents/sandbox
```



1 .env 配置 .env

2 安装依赖

3 agentseek dev

隔离执行 Shell · 文件读写 · 流式工具调用 · 优雅退出自动清理

安全与可观测：形成闭环

任意一层都不足够；让限制、审批、Trace 与产物审查互相补位

默认原则：沙箱输出也是不可信输入；不得直接执行、发布或当作可靠事实。

生产就绪检查表

把“能跑”升级为“可控、可审查、可恢查”的执行系统

✓	**运行环境**	Provider 的区域、镜像、依赖与默认工作目录已确认
✓	**生命周期**	Thread / Assistant 作用域、TTL、快照和删除策略已定义
✓	**文件边界**	输入由 upload_files 审查；产物经 download_files 复核
✓	**安全策略**	秘密不入沙箱；网络、HITL、最小权限与出站监控已配置
✓	**可观测性**	Trace 可还原工具调用；异常输出与产物有处置路径

下一篇：文件权限 — 用声明式规则约束 Agent 的文件操作。