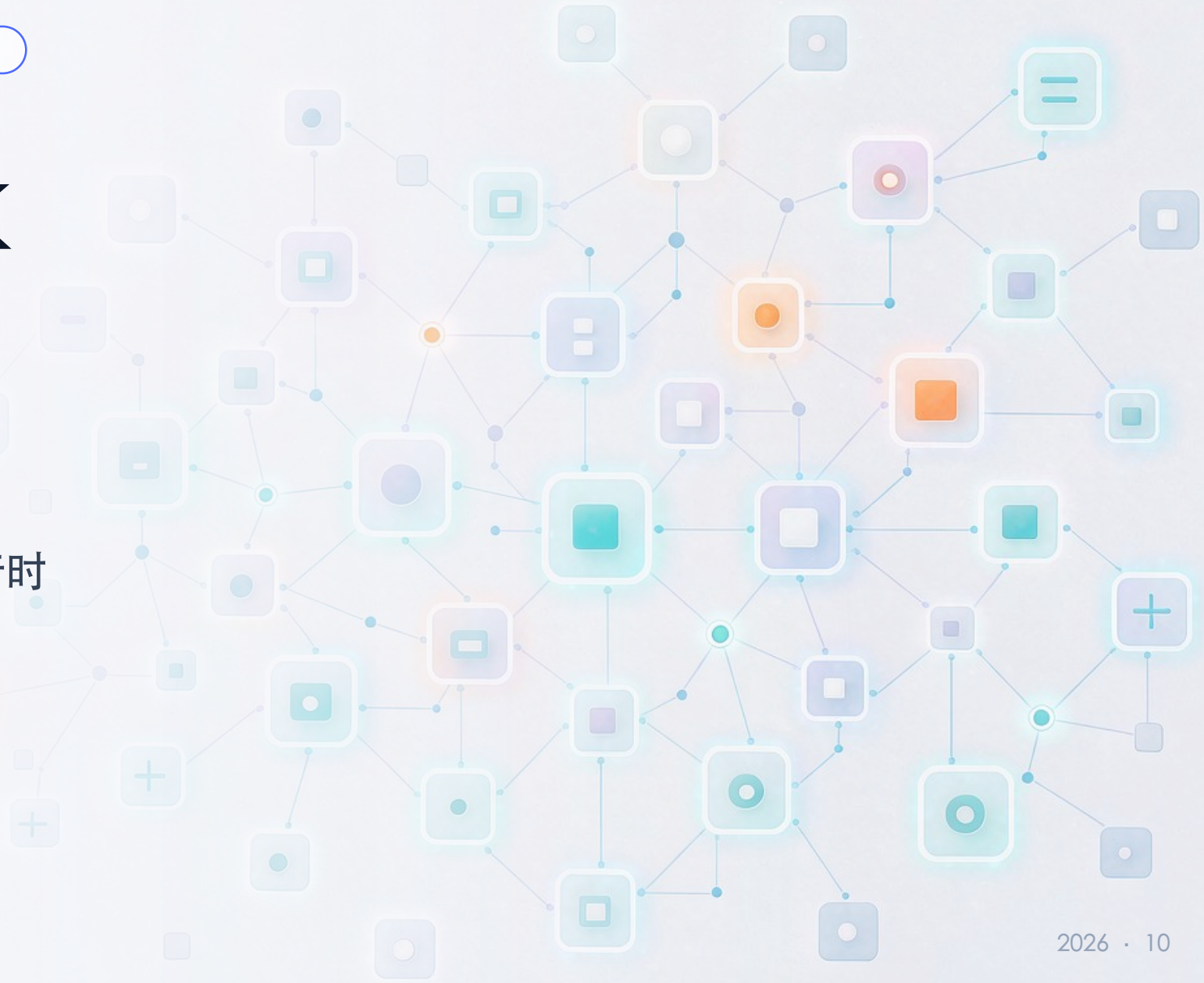
OPEN SOURCE · 开发者预览版 · MIT LICENSE

DeepSeek Harness

万物皆插件的开源 Agent 运行时

Agent = Model + Harness



github.com/deepseek-ai/DeepSeek-Harness

2026 · 10

本次分享的三站

3

三站，一条认知路径

先立「是什么」——模型之外到底缺了什么；再拆「凭什么」——它的架构差异在哪里；最后划清「能干到哪」——能力、安全与上手路径。

01

重新认识它

它不是新模型，也不是训练框架——为什么一个只会输出的模型，需要一层「运行时」才能真正干活。

02

架构解剖

一切皆插件： Cordis 微内核如何让模型接入、工具、会话、沙箱全部变成可替换、可组合的组件。

03

能力边界

让它干活，同时把它关进笼子——三档权限、系统级沙箱、可回放的会话日志，以及五分钟上手路径。

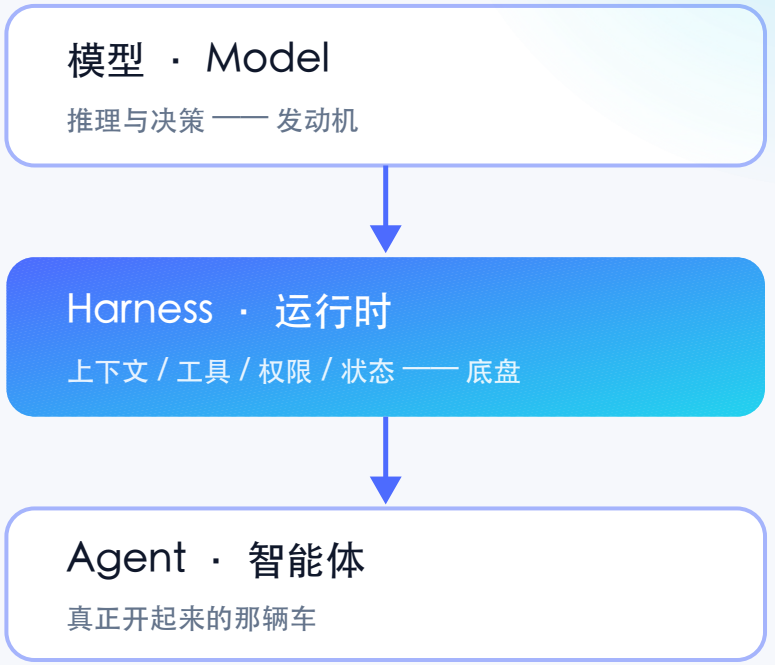
SECTION 01 / 03

01

重新认识它

「它不是新模型，是让模型真正干活的底盘。」

第一章先回答一个最常被问错的问题： DeepSeek Harness 到底是不是 DeepSeek 的新模型？答案藏在一条官方公式里。



一条公式，说清它的位置

OFFICIAL FORMULA · 官方极简定义

Agent =

Model + Harness

模型负责「想」

理解任务、拆解步骤、决定下一步调用什么——但它的输出只有文本，碰不到你的文件系统和终端。

只有推理 · 无执行

Harness 负责「做」

装配上下文、暴露工具、执行调用、记录会话、施加审批与沙箱——把想法落地成受约束的真实动作。

执行 · 记录 · 约束

判断：这条公式是理解全场的钥匙—— DSH 不是「又一个新模型」，而是把模型变成智能体的那层工程骨架。

有它和没它，差在哪



只有模型时

它只能告诉你「代码大概该怎么改」——输出一段建议，读不了仓库、跑不了测试、也验证不了结果。

交付一段话

装上 Harness 后

读仓库、搜代码、改实现、跑测试、看报错、再改一次——在授权范围内对真实环境动手，直到任务闭环。

交付一件做完的事

SO WHAT

Harness 交付的不是建议，而是一个闭环：同样的聪明大脑，从此有了手、脚和记忆。

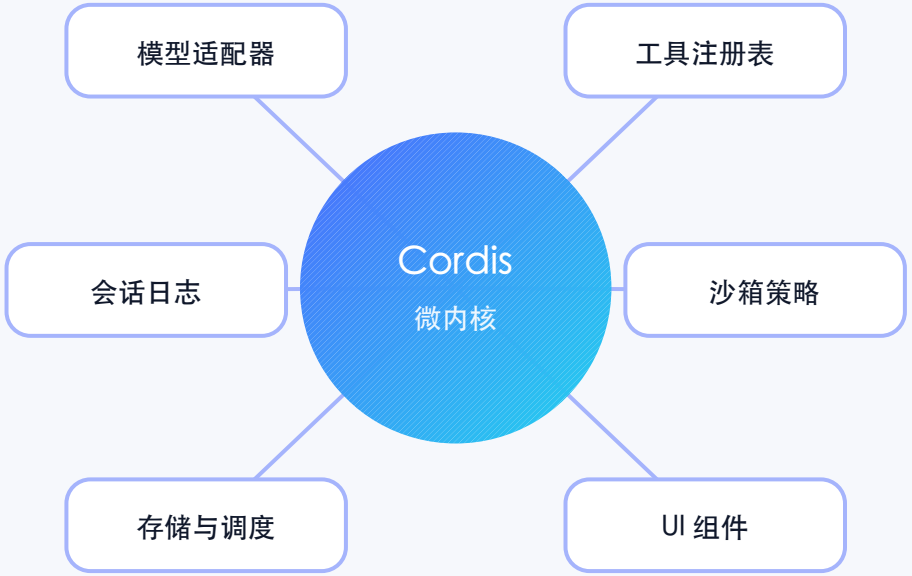
SECTION 02 / 03

02

架构解剖

「没有需要修补的特权核心。」

第二章拆开它的架构：为什么说 DSH 与同类工具的根本差异，在于从模型接入到 UI 的每一层都是插件。



一切皆插件：Cordis 微内核



微内核 Cordis

出自 Koishi 生态的插件内核，设计思想源自北京大学与 DeepSeek 的组合性研究论文——内核只负责装配与通信，业务能力全部下沉为插件，可按需动态组合。

没有特权核心

官方文档原话：扩展 Harness 只需在旁边挂载一个新插件，不需要给内核打补丁。加一个工具、换一家模型、改一套审批策略，本质上是同一种操作。

与同类工具的差异

Claude Code 、 Codex 把核心逻辑写死，只能在预留接口上有限定制； DSH 反其道而行——从模型到工具注册表，从会话日志到审批策略，全部插件化。

四种预设模式：先定岗位，再开跑

标准模式
STANDARD

全功能默认档：文件编辑、Shell 、检索、 Skills 、计划模式、子代理与工作流全部就绪。拿不准选哪个时就选它——能力最完整，兼容绝大多数编码与办公任务。

全功能 · 默认推荐

代码模式
CODE MODE · PTC

把工具包成一套 TypeScript SDK： 模型不再逐次调用函数，而是写一段小程序把多步操作一次执行完，减少往返——批量任务的效率可提升 3–8 倍。

批量任务 · 提效 3–8×

极简模式
MINIMAL

只保留 bash 与 str_replace_editor 两个工具，禁用其余插件。工具越少、变量越少，结果越可比——专用于性能评测和轻量精准调试。

评测 · 轻量调试

创造模式
CREATOR

继承标准模式的全部能力，追加运行时内省、插件热加载与自定义 Agent 模板——用于开发插件、调试内核，把团队私有工作流沉淀成可复用预设。

插件开发 · 定制

注意：预设决定工具集与规则，新建会话时选定、运行中不可切换——选模式不是调开关，而是给 Agent 分岗位。

热度，是最直接的投票

200,000+

GitHub Stars · 开源约三周

24,000+ Forks

上线数小时破 33,000 星

MIT 协议

数据截至 2026 年 9 月 · deepseek-ai/DeepSeek-Harness

这意味着什么

被 star 的可能只是好奇心，被 fork 的一定是改造欲——社区在把 DSH 改装成自己的运行时，官方仓库之外，社区插件生态已颇具规模。

SO WHAT

判断：预览版迭代极快，官方明示会出现破坏性变更——生产环境必须锁定版本，并在升级前备份会话数据。

SECTION 03 / 03

03

能力边界



「让它干活，同时把它关进笼子。」

第三章讲能力与边界：它能替你跑命令、改文件，那边界画在哪里？做过的事能不能追溯？最后，用五分钟把它跑起来。

三档权限，一层真沙箱



权限限制的是「写」，不是「看」——手被绑住了，眼睛是自由的。

只读 · Read Only

只看不改，任何文件都不会被修改——适合调查、总结、出方案。

工作区写 · Workspace Write

只能在划定的工作区内写文件——日常默认档，先把范围圈死。

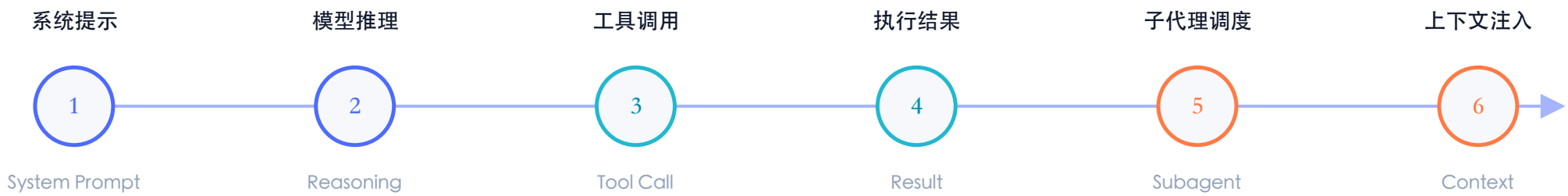
全访问 · danger-full-access

全盘读写、无边界。内部名里的 danger 已说明它是「危险模式」而非「高级模式」，切换前会二次确认。

底层是操作系统级沙箱

Linux Landlock / bwrap 、 macOS Seatbelt 、 Windows ACL 受限令牌；敏感操作触发审批弹窗，绝不静默放行。

一切都被记下来： append-only 日志



只追加 · 不可篡改 · 单一事件流

可恢复 Resume

会话自动永久保存：关掉浏览器、换台机器，下次接着上次的进度继续跑，不丢上下文。

可分支 Fork

从任意事件分叉出新会话——同一个任务试三条路，互不污染主线，随时回头。

可回放 Replay

轨迹面板逐条回溯模型思考、工具调用与文件修改——报错能定位，过程可审计。

底部状态栏实时显示当前模型、上下文 Token 占用、推理速度 TPS 与缓存命中率——可观测性是内建的，不是外加的。

五分钟，把它跑起来

```
$ npx @deepseek-ai/dsh web
# 一行启动：自动拉包，打开浏览器
→ http://127.0.0.1:3080

$ npm install -g @deepseek-ai/dsh
# 日常使用：全局安装后直接 dsh 命令
$ dsh web --port 8080
# 3080 被占用时换一个端口
```

ENVIRONMENT · 环境要求

Node.js ⩾ 22.19 （推荐 24 LTS） · Windows / macOS / Linux · 内存 ⩾ 4G

01 配置模型

Settings → Models 粘贴 API Key：官方 Key 在 platform.deepseek.com 创建，也支持 OpenAI 兼容提供商；密钥只写不回显，存于 .credentials.yaml 。

02 选定工作区

首页 Choose workspace 选中目录后，输入框才会解锁——「输入框是灰的」是新手第一道坎。别选系统盘根目录，建议建一个空白专属文件夹。

03 选预设模式，开跑

新建会话时选定四种预设之一，先丢给它一个只读任务验证链路——能正确列出工作区目录结构，就说明部署完成。

两条容易误会的「设计」：终端窗口不能关（Host 进程在那里）；默认只监听 127.0.0.1，局域网访问不到。

TAKEAWAY · 一句话带走

模型是引擎，Harness 是底盘；

真正开起来的那辆车，才叫 Agent。

Agent = Model + Harness