D

DeepSeek Harness

dsh · developer preview

# DeepSeek Harness

## DeepSeek 开源的智能体（Agent）运行框架

模型是灵魂， Harness 是让它看得见、动得了、跑得久的工程外壳。

v0.2.0–rc.2 | MIT 开源 | 本地优先 | 一切皆插件

内容基于 dsh 0.2.0–rc.2 的实际安装包与公开资料整理

AGENDA

# 今天讲什么




01 **认识 DeepSeek Harness**
它是什么，解决什么问题

02 **为什么需要 Harness**
模型之外的工程外壳

03 **设计哲学**
一切皆插件，装卸皆热插拔

04 **架构总览**
Cordis 微内核与分层组合

05 **四种内置模式**
标准 / PTC / 极简 / 创造

06 **能力全景**
工具矩阵与 Trajectory 轨迹

07 **安全与权限**
沙箱、审批与权限预设

08 **生态与上手**
插件生态与快速开始

WHAT IS IT

# DeepSeek Harness 是什么



DeepSeek Harness （命令名 dsh ）是 DeepSeek 开源的智能体运行框架。它把模型之外的一切工程能力——读写文件、执行命令、调用工具、管理上下文、失败重试、长时间连续工作——打包成一套可插拔、可观测、可替换的运行时。

### 一个本地编码 Agent

你在桌面端或浏览器里提需求，它在你自己机器的工作目录里读代码、改代码、跑命令，每一次工具调用都完整展示在时间线上。

### 一个可插拔的运行时

模型、工具、技能、沙箱、存储、 UI 全部以插件形式接入。装插件、卸插件、换整套界面，都不需要重启。

### 一个可回放的会话黑匣子

模型看到的系统提示、推理过程、工具调用与结果、子代理调度、上下文注入，全部写入只追加的事件流，随时可查。

WHY IT MATTERS

# 为什么需要 Harness



## 模型层 灵魂

- 理解意图、拆解任务、推理规划
- 生成代码、文本与结构化输出
- 判断下一步该做什么、什么时候停
- 决定调用哪个工具、按什么顺序
- 本身看不见文件，也碰不到终端
- 能力上限由模型参数本身决定

## Harness 层 身体

- 读文件、写文件、按模式查找、正则搜索
- 执行 shell 命令，前台与后台长任务
- 管理上下文、压缩历史、保留要点
- 失败重试、断点续跑、长时间连续作业
- 调度子代理与多代理工作流
- 权限审批、沙箱隔离、操作可审计

同一个模型，换一套 Harness，实际表现可以差出一个量级——**模型决定上限，Harness 决定能不能落地。**

DESIGN PHILOSOPHY

# 一切皆插件



DSH 的口号是「一切皆插件」。系统里没有特权模块——**连默认 UI 、模型接入、沙箱实现都是插件。** 每个插件的注册副作用都被完整追踪，卸载时自动回收，因此装卸都可以在运行时完成。

| | | | |
|---|---|---|---|
| 模型接入 | 工具 Tools | 技能 Skills | Agent 循环 |
| 会话与持久化 | 上下文压缩 | 沙箱 Sandbox | 存储 Storage |
| 定时调度 | UI 界面 | 权限与审批 | 遥测与日志 |

社区常把它类比成 Minecraft：**本体是原版生存，插件是 Mod ，插件目录站是 CurseForge 。**内核叫 Cordis ，从 Koishi 社区抽出的微内核，提供依赖注入、服务与事件、可逆副作用。

ARCHITECTURE

# 架构总览：微内核 + 分层组合



| | |
|---|---|
| 入口层 | 桌面应用 · Web（浏览器） · CLI · Headless 无界面 · SDK（JSON-RPC） · ACP |
| 内核层 | Cordis 微内核：插件加载器 · 依赖注入 · 服务与事件 · 可逆副作用 · 配置 patch 叠加与热重载 |
| 能力层 | Agent 循环 · 工具集 · 技能 · 会话与持久化 · 沙箱 · 存储 · 定时调度 · 上下文压缩 · 遥测 |
| 模型层 | DeepSeek 模型与平台账号 · API Key 接入 · 兼容第三方模型 —— 同样以插件形式挂载 |

配置树以空根为起点，按「组合包 patch → profile patch → home patch → 命令行覆盖」的顺序叠加。

BUILT-IN MODES

# 四种内置模式，按任务切换



## 标准
Standard

- 全功能版本
- 文件编辑与 Shell
- 文件 / 网页搜索
- 技能、计划、目标
- 子代理与工作流
- 适合日常开发

## PTC
Programmatic Tool Calling

- 工具经 SDK 暴露给模型
- 模型直接写程序调用工具
- 十轮往返压成一次执行
- 减少往返与上下文开销
- 适合批量、重复操作

## 极简
Minimal

- 只保留两个工具
- 持久化 bash
- 文件编辑器
- 剥掉外壳的全部花活
- 为模型基准测试而设计

## 创造
Creator

- 用来创造新模式
- 在 Cordis YAML 中定义
- 新的 Agent 子插件组合
- 声明即生效
- 改的是之后新建的会话

模式本质上是 Agent 的子插件清单（preset）：换个 preset，就换了一套工具、提示词与工作方式。

CAPABILITIES

# 能力全景：工具矩阵



## 文件与代码

- 读取 / 写入 / 精确替换
- 按模式查找文件、正则搜索内容
- 改动前先观察、改动后可回看

## 终端与进程

- 前台与后台 shell 命令
- 持久化会话型终端
- 作业管理、超时与输出回收

## 检索与研究

- 网页搜索与结果摘要
- 抓取并解析指定 URL
- 外部网页内容按不可信数据处理

## 协作与编排

- 子代理：独立上下文处理子任务，或分叉继承当前对话
- 多代理工作流：用脚本把任务扇出到大量子代理，分阶段汇总
- 待办清单与持久目标：跨轮次自动续跑，直到目标真正达成

## 交付与呈现

- 办公文档： Word 、 Excel 、 PowerPoint 的创建、编辑与校验
- PDF 转换、文件级交付卡片、图像与表格结果
- 技能系统 Skills ： 把可复用的任务指令沉淀成按需加载的模块

# Trajectory： 把 Agent 的一举一动摊开



## 只追加的事件流

模型看到的每一样东西——系统提示词、推理过程、工具调用与返回结果、子代理调度、每一次上下文注入——都按来源记录在同一条只追加的会话日志里。

会话恢复、分叉、检索与回放，全部建立在同一条事件流上。出问题时，你能完整复盘它当时看到了什么、为什么这么做。

基于同一条事件流

恢复会话　分叉试验　全文检索　回放复盘

轨迹视图可以按来源逐条过滤，相当于给每一次 Agent 运行装了一个内置黑匣子。

一次运行的典型事件序列

| | |
|---|---|
| 系统提示词 | 指令、工具说明、技能目录 |
| 用户消息 | 需求 + 引用的文件与上下文 |
| 模型推理 | 思考过程、计划与下一步判断 |
| 工具调用 | 读取、编辑、执行命令、搜索 |
| 工具结果 | 输出、报错、被裁剪或落盘的内容 |
| 子代理调度 | 独立上下文并行处理子任务 |
| 上下文注入 | 时间、环境、目标、记忆与提醒 |

SAFETY & PERMISSIONS

# 安全与权限：能力越大，边界越要清楚



## 文件沙箱

文件策略　工作区可写

文件策略按会话声明可写范围，所有路径在解析后校验：越界访问被明确拒绝并回报，不会静默忽略，也不会悄悄降级成只读。

## 命令沙箱

进程隔离　可替换实现

命令在受约束的环境中执行，可限制可写目录与网络访问；沙箱本身也是插件，可以按平台与需求替换成别的隔离实现。

## 审批策略

人工把关　可见可拒

需要更高权限的操作会主动请求批准，并把完整命令展示给你；如果没有可用的审批通道，默认失败关闭，而不是直接放行。

**权限预设**

按任务风险选择放手程度

只读　工作区可写　完全访问　需要更高权限时逐次批准

ECOSYSTEM

# 插件生态：发布一天就长出了一片森林



GitHub topic

**dsh-plugin**

任何打上该标签的仓库都可被发现

社区收录

**近 300 个仓库**

dsh 发布后 24 小时内的统计数字

装卸 20+ 插件

**0 次重启**

热插拔由 Cordis 的可逆副作用保证

## 社区已经做出来的东西，说明了「一切皆插件」能走多远

- **dsh-plan-execute** 规划与执行用不同模型，自动路由
- **dsh-vision** 给纯文本模型补上读图能力
- **sandbox-micro / mxc / nono** 三种隔离方案任选其一
- **QQ / 微信 / 飞书 / Telegram** 变成群里可 @ 的机器人
- **dsh-cc-tui · opencode-server** 换一套终端界面或前端
- **dsh-qq2006 · dsh-ui-whale** 2006 年 QQ 皮肤、像素鲸鱼

插件按声明的 dsh 版本范围做兼容性校验，不兼容的插件需要显式确认豁免才会安装。

SCENARIOS

# 典型应用场景



## 功能开发与重构

把需求丢给它：读代码、改文件、跑测试、修报错。改完之后你可以逐条回看它做了什么。

## 代码审查与缺陷排查

用只读或工作区可写权限扫描项目、定位问题、给出按严重程度排序的结论，并保留完整的取证过程。

## 数据处理与办公交付

直接产出 Word 、 Excel 、 PowerPoint 、 PDF 等交付物，脚本、数据与文档在同一次会话里完成。

## 长任务与研究

用持久目标加后台作业加子代理：一次交代，它跨轮次自动续跑，直到目标真正达成。

共同点：工作目录就是你自己的项目目录，所有动作都在本地发生，并留下可审计的记录。

# 快速上手



1

## 选一种载体

桌面应用开箱即用；也可以一条命令起 Web 端，在浏览器里用。

2

## 把工作目录交给它

启动目录即默认工作区根目录；一个工作区对应一个本地项目目录。

3

## 选模式、说需求

日常开发用标准模式；批量重复操作用 PTC；跑基准用极简。

**常用命令**

```
# 一条命令在浏览器里启动 Web 端
npx @deepseek-ai/dsh web

# 桌面应用自带的 CLI
dsh web

# 无界面跑一个任务，打印答案后退出
dsh --profile headless "运行测试"

# 给某个 profile 管理插件
dsh plugin --profile web add <pkg>
```

dsh 是唯一受支持的启动器；web、headless、sdk、acp 等都是 profile，首次使用时自动初始化。

TAKEAWAYS

# 小结



### Harness 是模型之外的那层工程外壳

读文件、调工具、管上下文、重试、长时间连续工作——它决定模型的能力能不能真正落到项目上。

### 一切皆插件，所以一切都可以替换

模型、工具、沙箱、UI 都是插件；可逆副作用让装卸都在运行时完成，不用重启。

### 全链路可观测，所以敢放手让它干

只追加的事件流记录模型看到的一切，配合沙箱与审批策略，把「能干活」和「可控」放在一起。

**模型是灵魂，Harness 是身体。** DeepSeek Harness 把身体做成了可插拔、可观测、可替换的一层——所以换模型、换工具、换界面，都只是换一个插件而已。