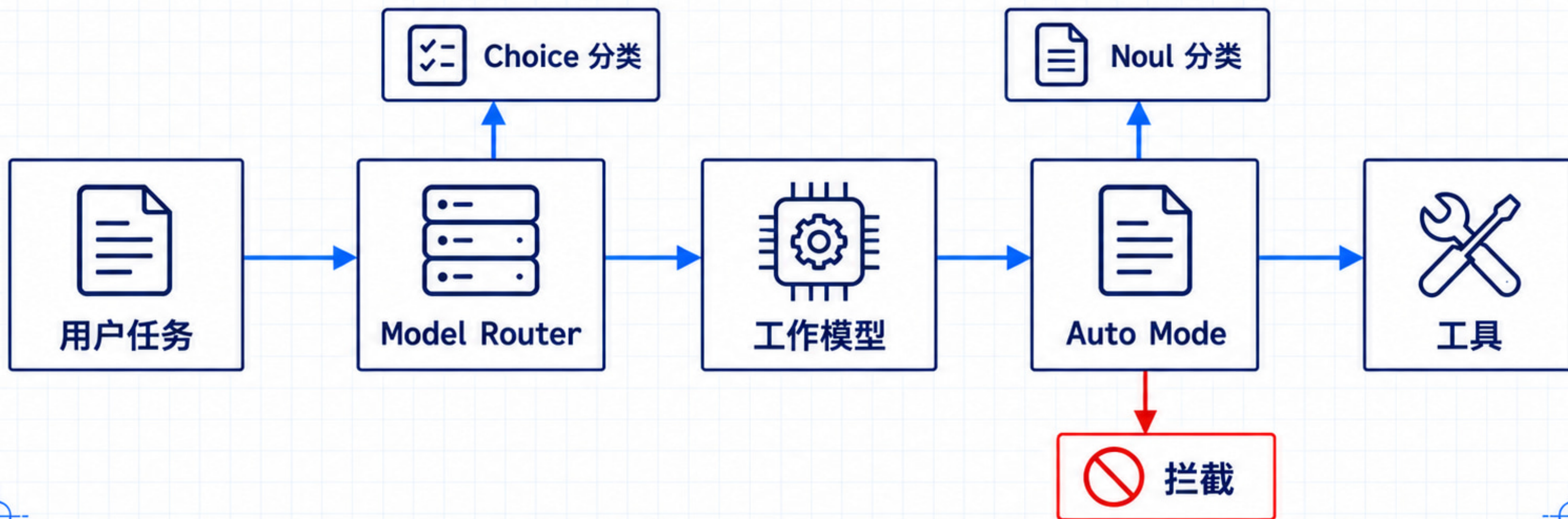


Deep Agents 实战

第 21 讲：Decision Model Harness

模型路由与 Auto Mode



Agent 有了工具，还要做两类判断

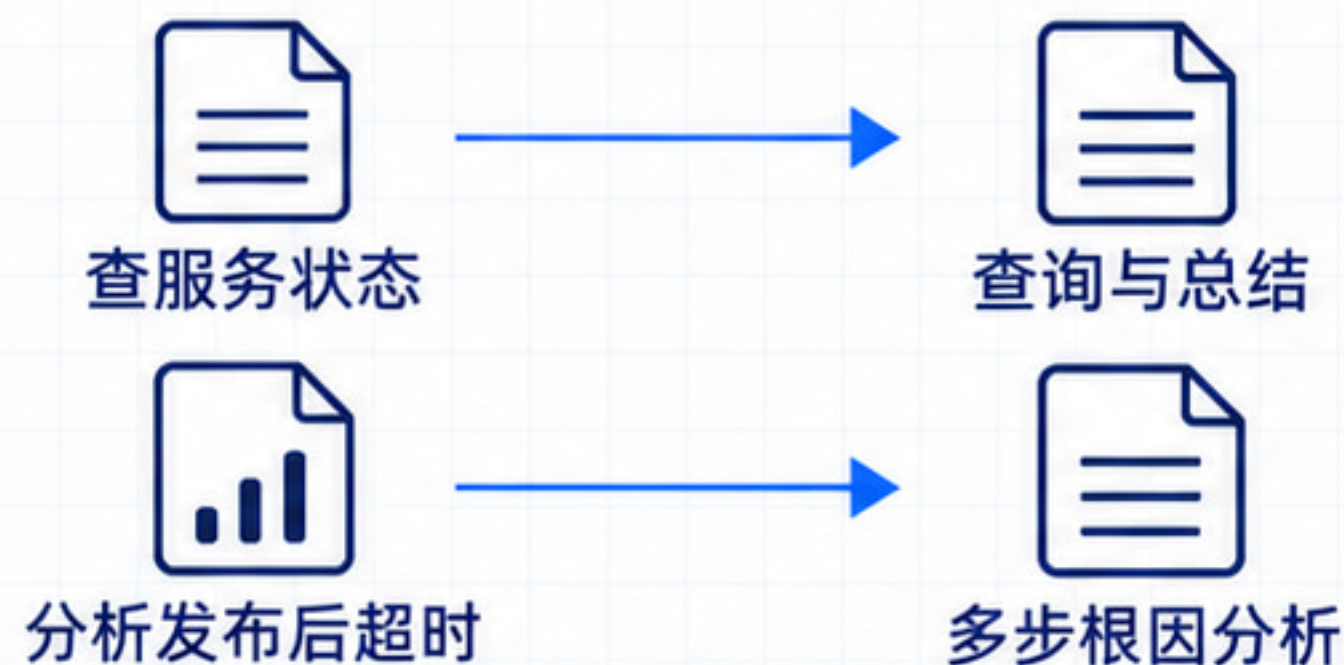
前面几讲：规划、工具、子 Agent → 本讲：模型选择与工具授权



示例：构建一个测试环境的运维 Agent

查询服务状态、分析故障，并按用户授权执行变更

问题 1：任务交给哪个模型？



能力要求不同，选择合适的工作模型

问题 2：工具提案能否执行？



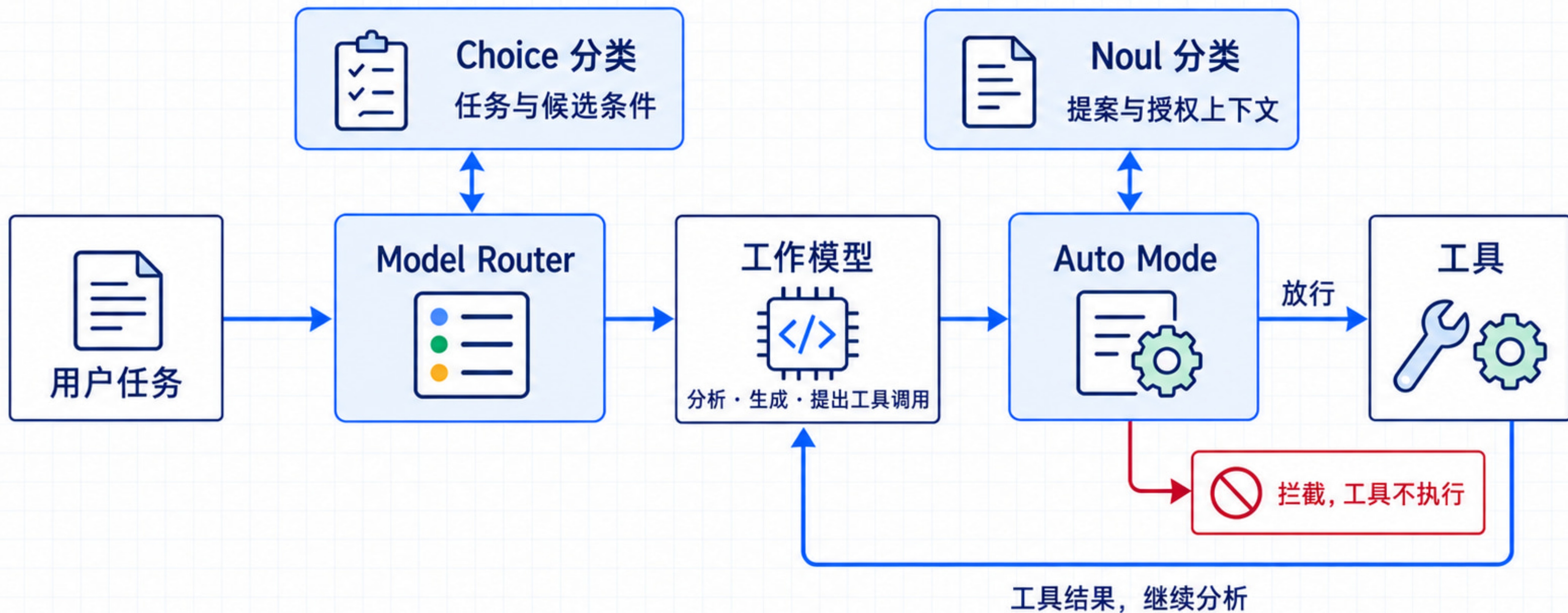
能分析故障，不代表获得变更授权



Model Router 选择模型 · Auto Mode 检查工具提案

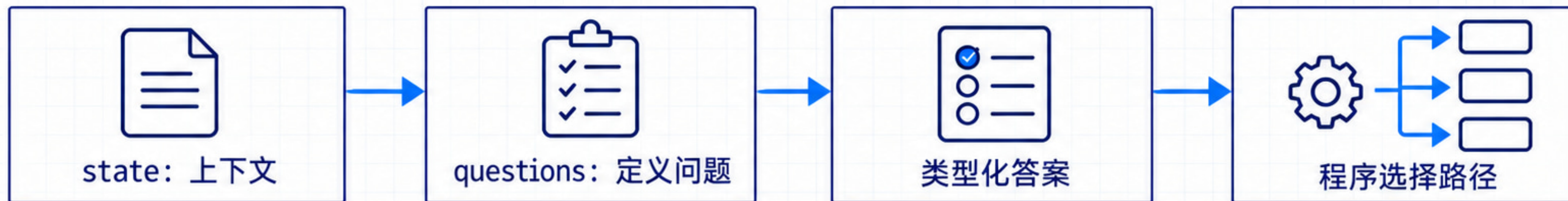
工作模型与决策模型的分工

工作模型负责生成与分析，决策模型回答定义好的问题



类型化决策接口，可连接不同后端

Jev 是实现之一；兼容服务复用 System One 契约



Choice



从给定候选中
选择一个

Noul



某个陈述
为真的概率

Score



输入处于哪一个
有序等级

同一契约，三种后端选择



TypeSafe / Jev



硅基流动 / Kev-4b



硅基流动 / SemIf

</> POST /v1/systemone · Jev-compatible 接口



接口兼容后，仍需重新评估判断效果与阈值

Model Router 的候选与条件

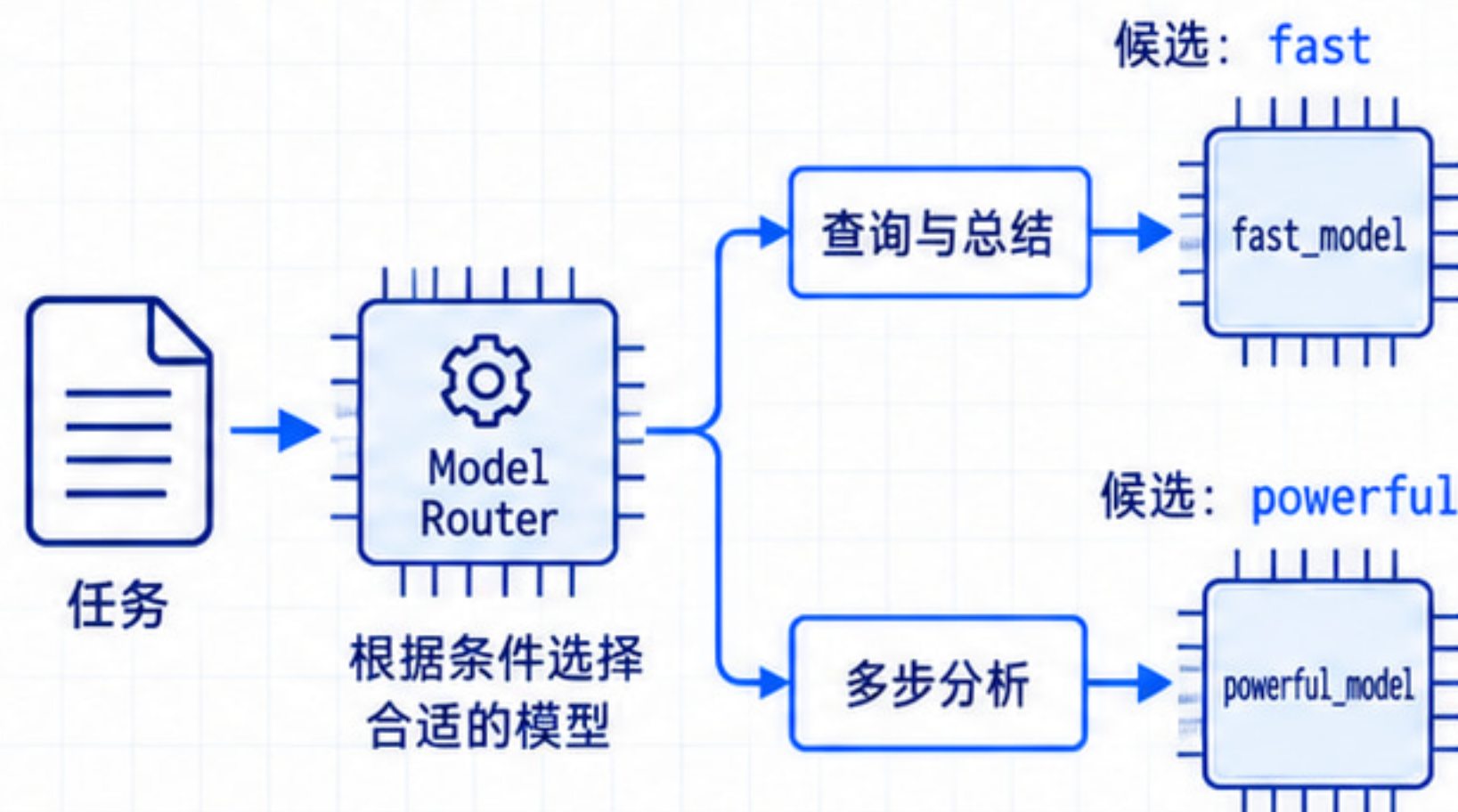
先评估模型，再把适合的任务写成 criteria

路由器配置示例

```
router = ModelRouterMiddleware(  
    choices={  
        "fast": ModelChoice(model=fast_model, criteria="查询与总结"),  
        "powerful": ModelChoice(model=powerful_model, criteria="多步分析"),  
    },  
    instructions="选择能完成任务的最低成本候选模型",  
)
```

fast_model 和 powerful_model 是已初始化的工作模型

任务路由流程

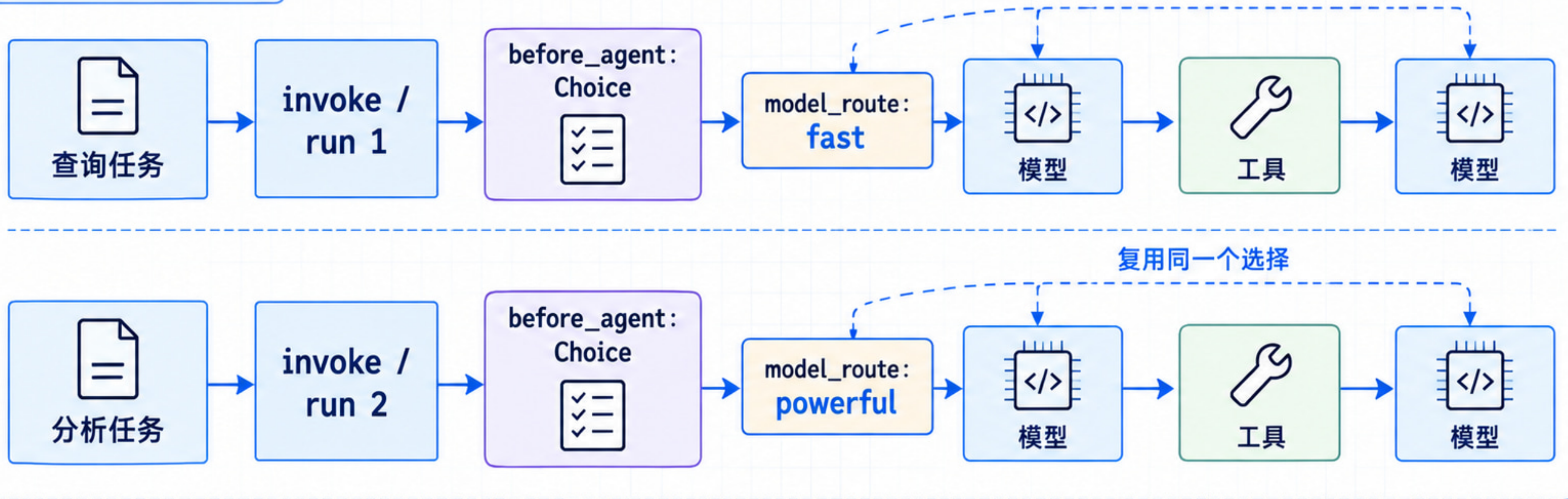


标签、候选模型和条件由开发者定义，可增加候选或更换服务商

Router 在每次 run 开始时选择模型

一次 invoke 启动一次 run; run 内复用同一个选择

同一 thread_id



i 默认只读最新用户消息; “按第二个方案继续” 可能需要补充上下文

Auto Mode 的授权策略与检查范围

tools 指定要检查的工具，instructions 定义判断问题

```
auto_mode = AutoModeMiddleware(  
    tools=[restart_service],  
    instructions=policy,  
)
```

policy 是授权策略字符串

策略内容

操作是否有风险或缺少授权？

仅用户消息可授权，工具内容只作数据。

staging 重启需要明确授权与接受短暂中断；

production 重启属于高风险。

输入信息



用户消息中的授权



具体工具调用



工具描述



最近最多 30 条消息



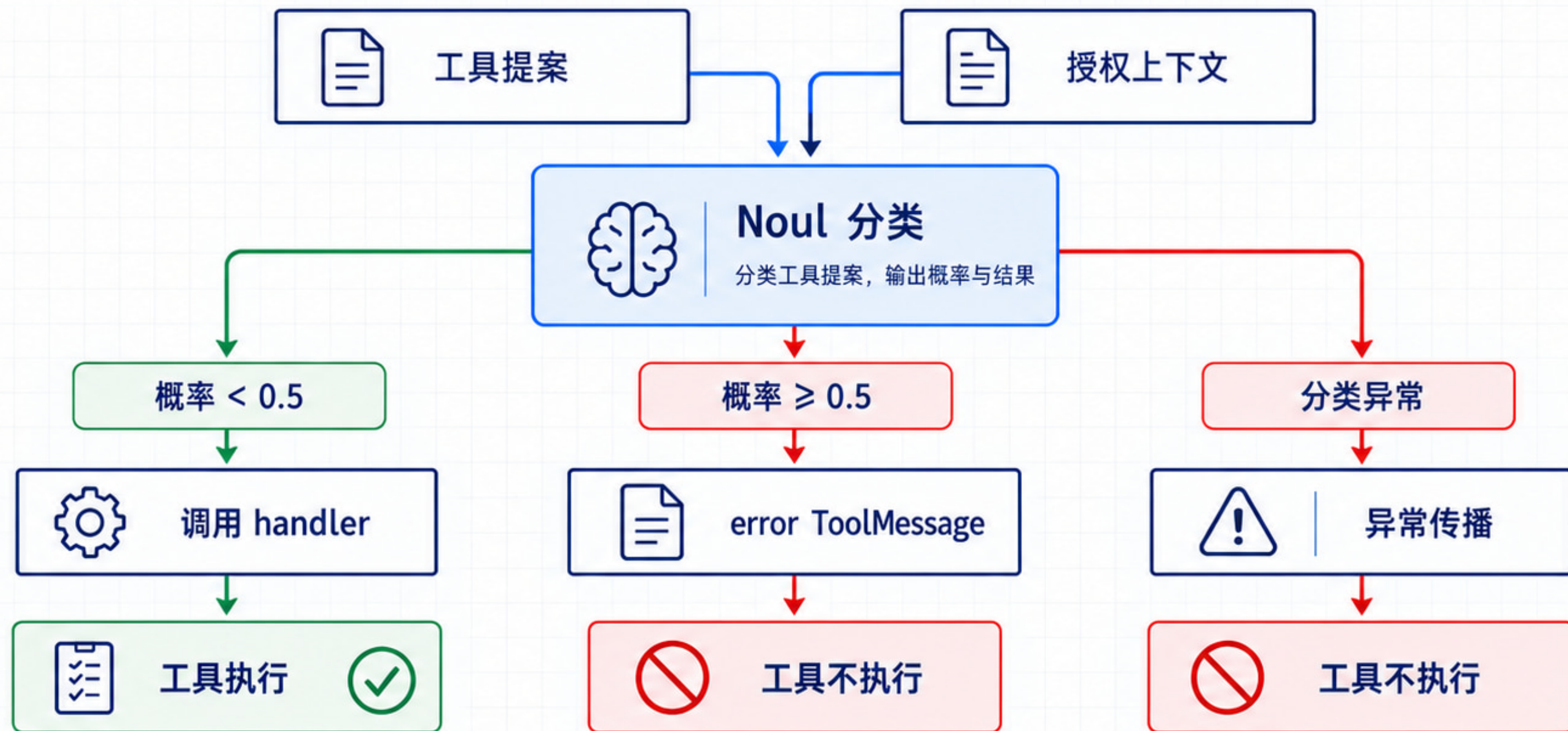
是否有风险
或缺少授权？



仅用户消息可授权；未列入 tools 的工具跳过分类

Auto Mode 的放行与拦截路径

在 wrap_tool_call 中，先分类，再决定是否调用 handler



拦截不会自动发起人工审批；审批与恢复运行需要 HITL

工作模型与决策后端分别配置

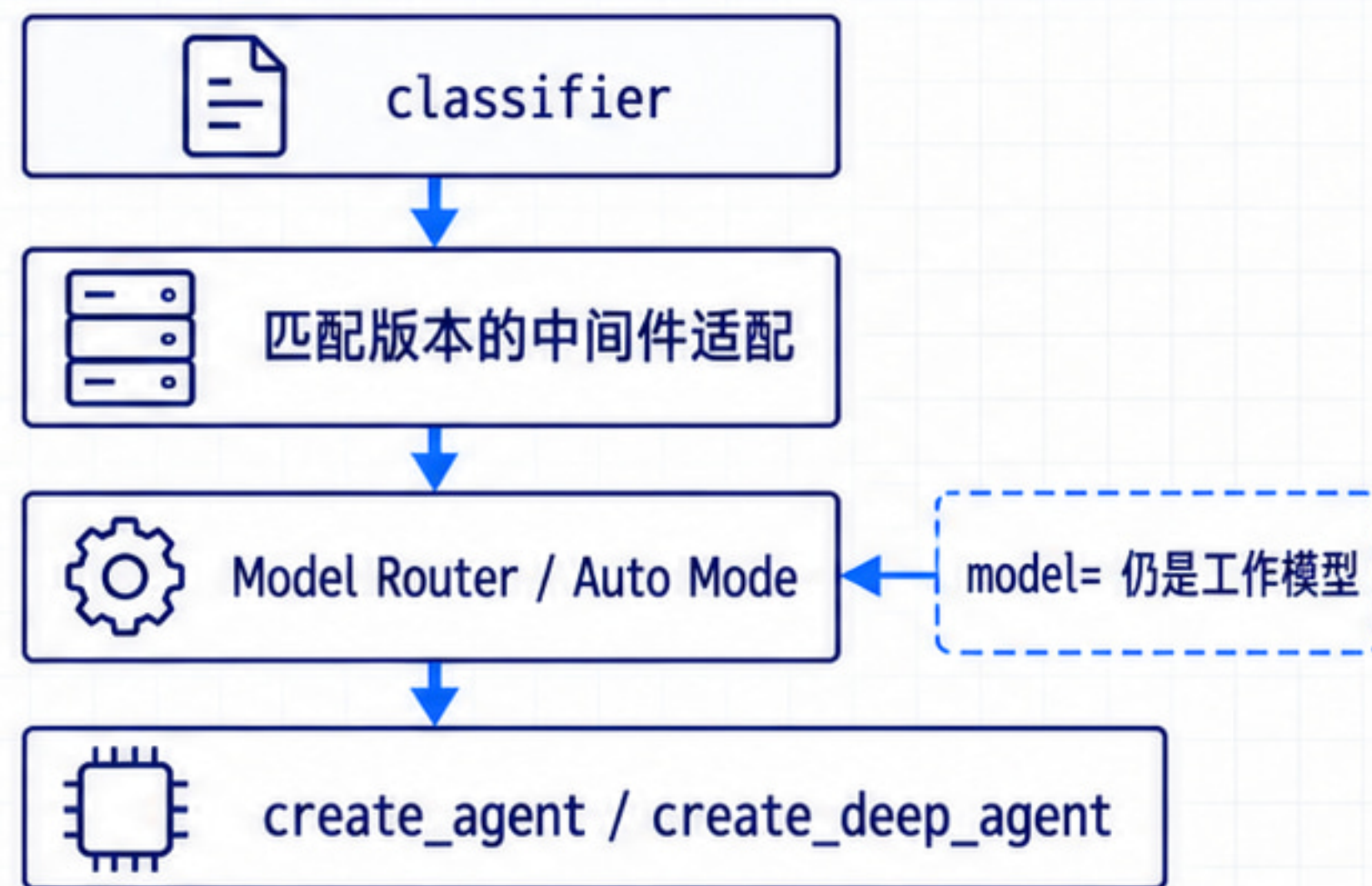
示例：连接硅基流动的 Jev-compatible 决策接口

1. 配置决策分类器

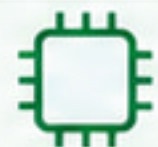
```
classifier = TypeSafeClassifier(  
    base_url="https://api.siliconflow.cn",  
    api_key=os.environ["SILICONFLOW_API_KEY"],  
    model="Kev-4b",  
)
```

示意片段：已导入 os 与 TypeSafeClassifier

2. 按中间件版本接入



本章 0.0.1a3 内置中间件无 classifier= 参数



仅初始化兼容分类器，不会替换内置中间件的决策后端

适用条件与收益评估

任务能力需求有差异，或执行许可依赖上下文

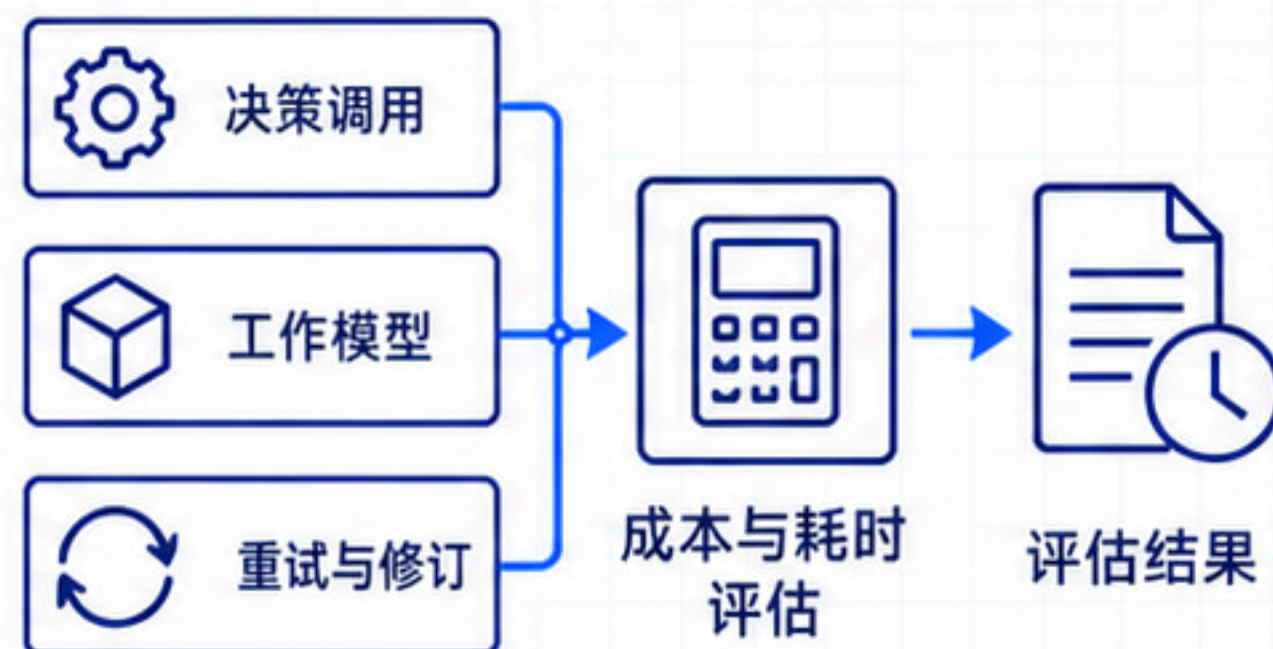
任务质量

事实、分析步骤、恢复计划是否达标



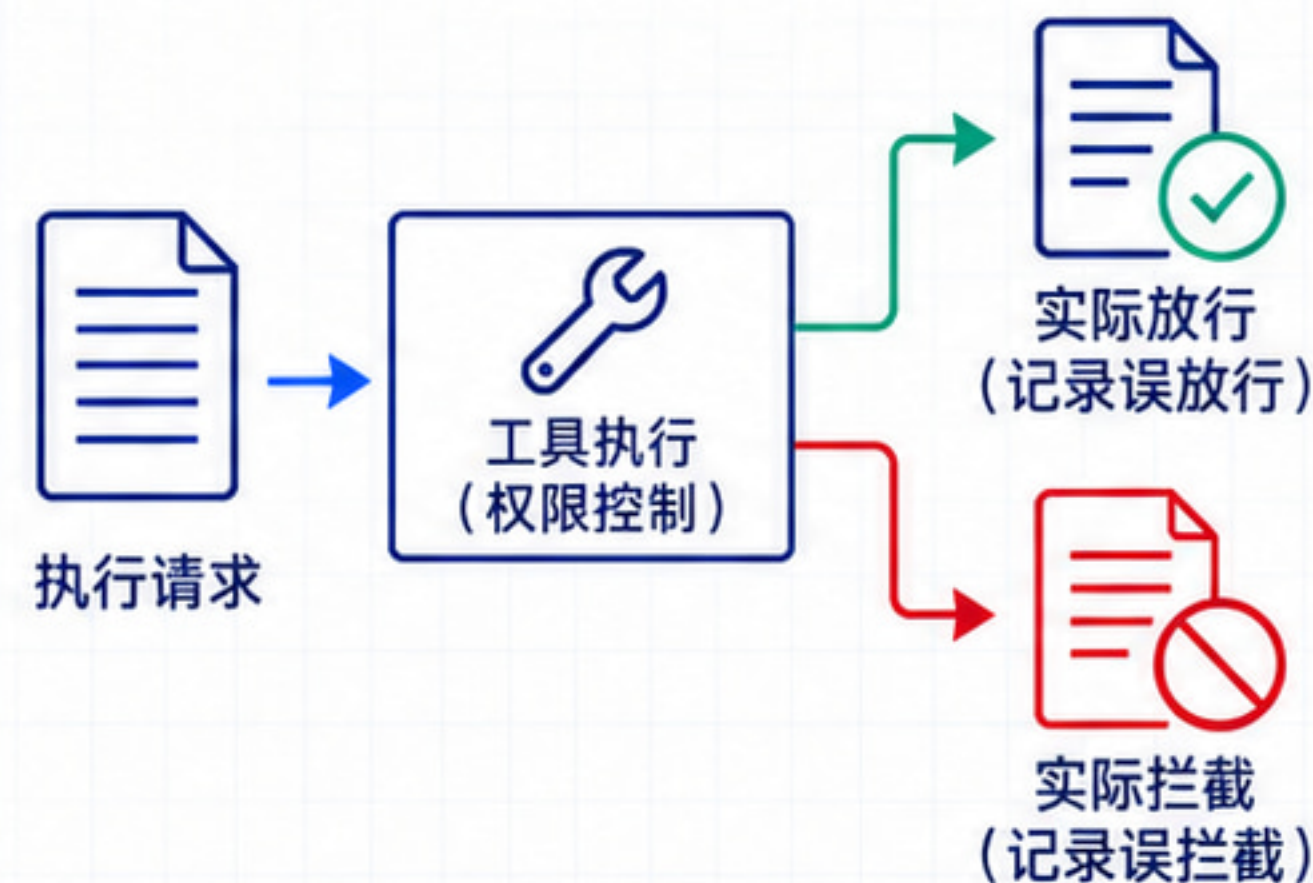
总成本与耗时

决策调用、工作模型、重试与修订



执行边界

核对工具实际执行，记录误放行与误拦截



所有任务同质时，路由收益可能有限；明确资源限制仍由工具与权限控制

AgentSeek 模板: Jev Harness 实验

安装 CLI, 生成项目, 观察两类决策如何生效

安装与拉取模板

1. 安装 AgentSeek

```
uv tool install --upgrade agentseek
```



2. 拉取 Jev Harness 模板

```
agentseek create langchain/jev-harness \  
--checkout main --no-input
```



`main` 获取当前模板; 生成目录为 `jev_harness_lab`

模板里可以实验什么



Model Router: 观察候选模型选择



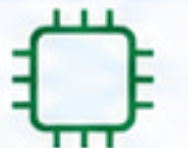
Auto Mode: 观察工具放行与拦截



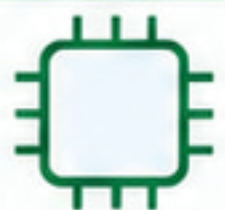
上下文对照: 同一提案, 不同授权



Arena: 对照不同决策后端



决策后端: **Jev / Kev-4b / SemIf**



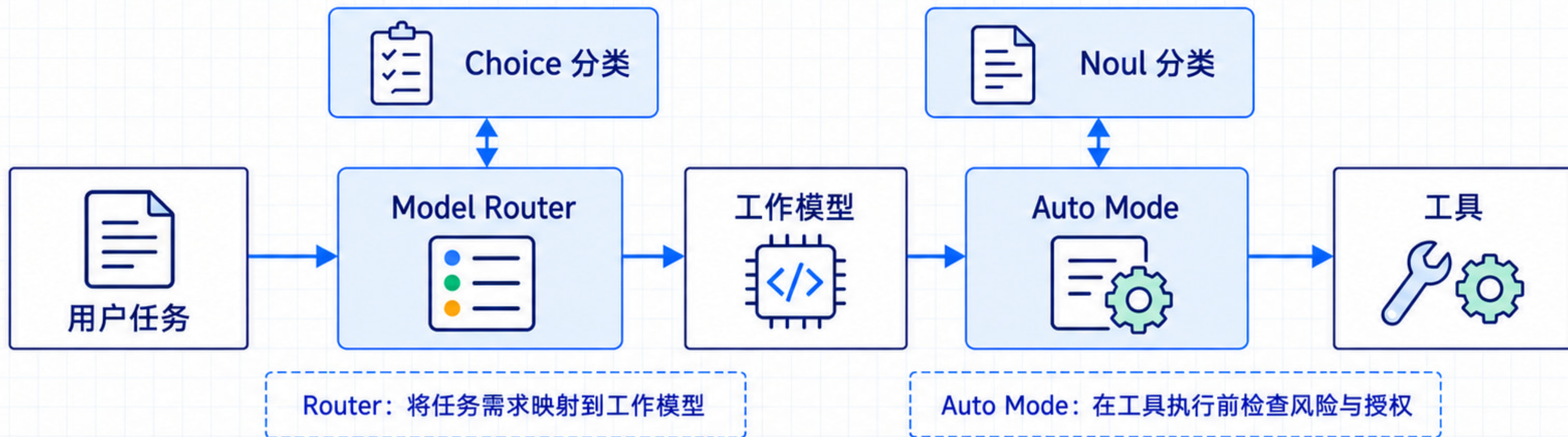
进入项目, 按 README 配置 Key 与依赖, 再用 `agentseek dev` 启动



工具为模拟操作; 决策模型负责判断, 工作模型负责回答

本章小结：任务分工与执行检查

Decision Model 将定义好的问题转成类型化答案



通过真实任务评估质量、
整体开销与执行边界

延伸阅读

- Building a Harness with Jev
- How to Build a Model Router in the Harness
- LangChain TypeSafe 集成文档