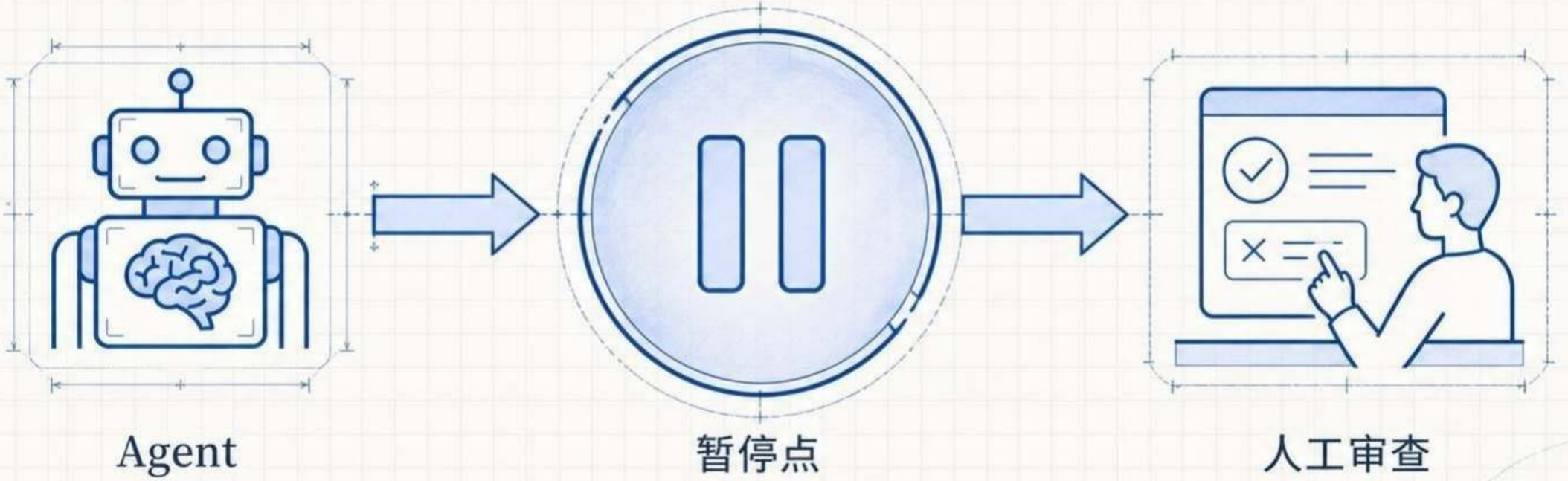
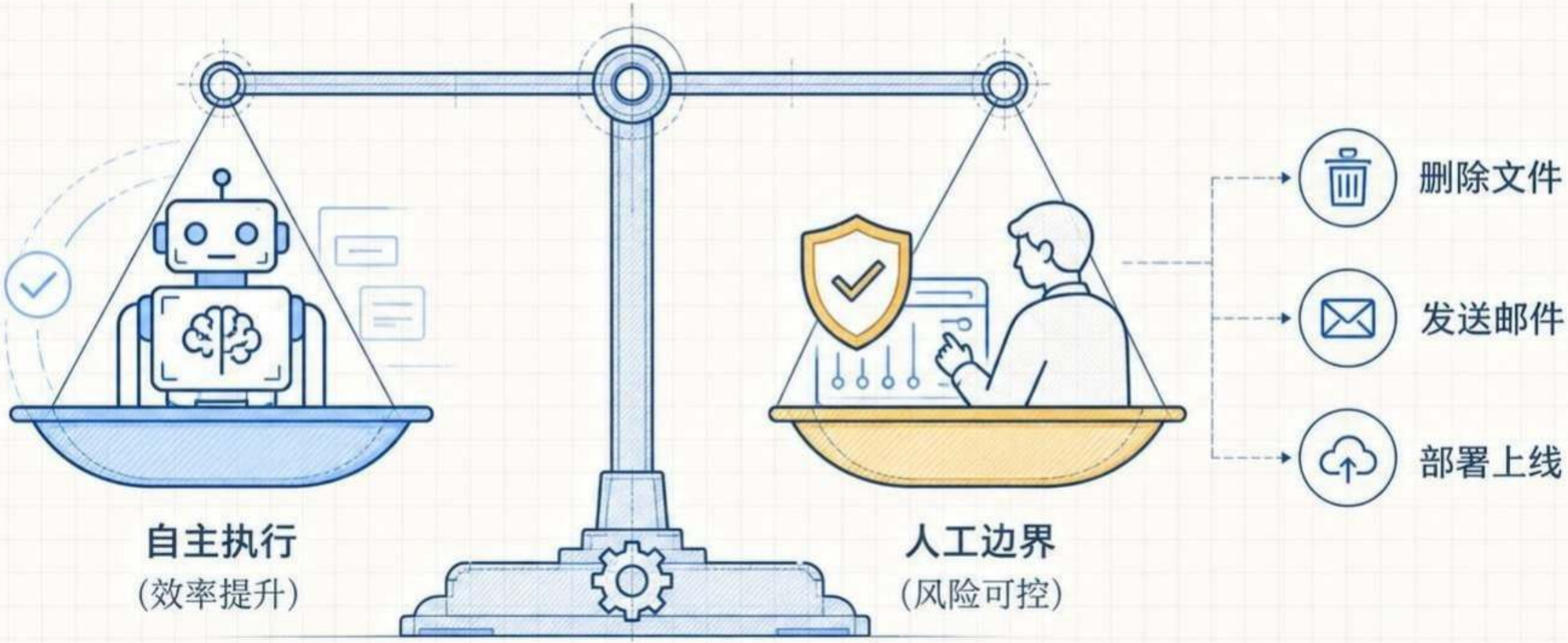
Deep Agents 实战

第 12 讲：HITL 基础 — 配置与决策



为什么需要 HITL？

自主性越高，越需要安全边界



最小可运行配置

先把 checkpointer 和 interrupt_on 接进 Agent

```python
checkpointer = MemorySaver()

agent = create_deep_agent(
    model=model,
    tools=[delete_file, read_file, send_email],
    interrupt_on={
        "delete_file": {"allowed_decisions": ["approve", "edit", "reject"]},
        "read_file": False,
        "send_email": {"allowed_decisions": ["approve", "reject"]},
    },
    checkpointer=checkpointer,
)
```

checkpointer：
保存可恢复状态

interrupt_on：
按工具配置是否打断

allowed_decisions：
限制人工可选动作

四种决策怎么配

不同工具开放不同人工动作

approve 通过

edit 修改

```
interrupt_on = {
    "delete_file": {"allowed_decisions": ["approve", "edit", "reject"]},
    "ask_user": {"allowed_decisions": ["respond"]},
    "read_file": False,
}
```

reject 拒绝

副作用工具：
只允许 approve / edit / reject

ask_user：
只开放 respond

只读工具：
无需中断

respond 不是拒绝
高风险用 reject

条件中断：按参数拦截

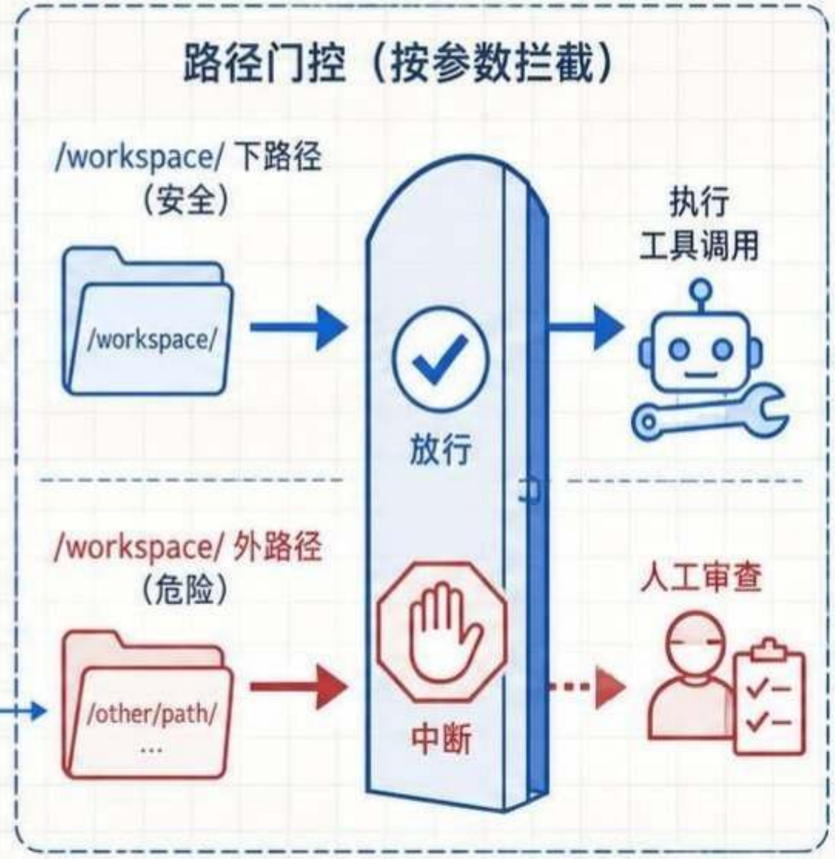
用 when 只审查真正危险的调用

```python
def writes_outside_workspace(request: ToolCallRequest) -> bool:
    path = request.tool_call["args"].get("file_path", "")
    return not path.startswith("/workspace/")


interrupt_on={
    "write_file": {
        "allowed_decisions": ["approve", "edit", "reject"],
        "when": writes_outside_workspace,
    }
}
```

ToolCallRequest:
读取工具参数

when:
只拦截命中的调用

/workspace/ 内路行，
外部路径中断



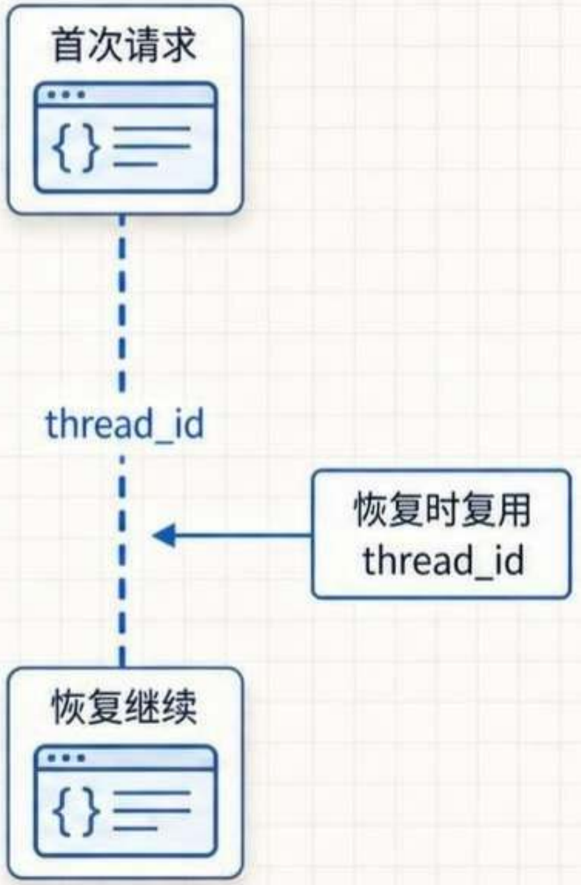
发起请求：带上 thread_id

恢复时必须回到同一条线程

生成唯一 thread_id

```python
import uuid

config = {"configurable": {"thread_id": str(uuid.uuid4())}}

result = agent.invoke(
    {"messages": [{"role": "user", "content": "删除 temp.txt"}]},
    config=config,
    version="v2",
)
```

请求携带同一个 config



读取中断内容

把待审工具展示给用户

```python
if result.interrupts:
    value = result.interrupts[0].value

    action_requests = value["action_requests"]
    review_configs = value["review_configs"]

for action in action_requests:
    args = action.get("arguments" ) or action.get("args", {})
    print(action["name"], args)
    # 展示 allowed_decisions 给用户
```

读取 interrupt value

提取 action_requests

提取 review_configs

兼容 arguments / args

展示给用户审批

待用户审批

1. 待审工具

write_file

向文件写入内容

2. 参数（arguments/args）

```
{
  "path": "/reports/summary.md",
  "content": "..."
}
```

3. 允许决策（allowed_decisions）

- approve
- reject

✓ 批准　　✕ 拒绝

resume： approve

批准原始工具调用，按 Agent 提出的参数执行

```
result = agent.invoke(
    Command(resume={
        "decisions": [
            {"type": "approve"}
        ]
    }),
    config=config,
    version="v2",
)
```

原始工具调用

write_file

向文件写入内容

原始参数（保持不变）

```
{
  "path": "/reports/summary.md",
  "content": "..."
}
```

1. **使用原始参数**
   不修改 Agent 提出的参数
2. **工具会真正执行**
   执行原始的工具调用
3. **适合确认无误的操作**
   如写文件、发送邮件等

用户批准
选择 approve

→

工具执行
按原始参数执行

→

Agent 继续
后续流程继续运行

resume：edit

先改参数，再执行工具

PYTHON

```python
decisions = [{
    "type": "edit",
    "edited_action": {
        "name": "send_email",
        "args": {
            "to": "team@example.com",
            "subject": "通知"
        }
    }
}]
```

```python
agent.invoke(Command(resume={"decisions": decisions}),
             config=config, version="v2")
```

原始参数

send_email	
to	team@example.com
subject	通知

修改后参数

send_email	
to	team@example.com
subject	通知

1. edited_action 必须包含 name
2. 修改后的参数放在 args
3. 只做保守修改

原始参数 → 人工修改 → 修改后执行

resume： reject

拒绝工具执行，并把原因反馈给 Agent

PYTHON

```
decisions = [{
    "type": "reject",
    "message": "用户拒绝删除。不要重试，
               请询问是否改为归档。"
}]

agent.invoke(
    Command(resume={"decisions": decisions}),
    config=config,
    version="v2",
)
```

拟执行工具

delete_file

args: {"path": "report.pdf"}

反馈给 Agent

用户拒绝删除。
不要重试，请询问是否改为归档。

1 工具不会执行

2 message 会返回给 Agent

3 用于删除、发送、部署等高风险动作

拒绝副作用工具时用 reject，不要用 respond

用户拒绝 → 返回反馈 → Agent 重新规划

resume： respond

人类直接作为工具返回值

PYTHON

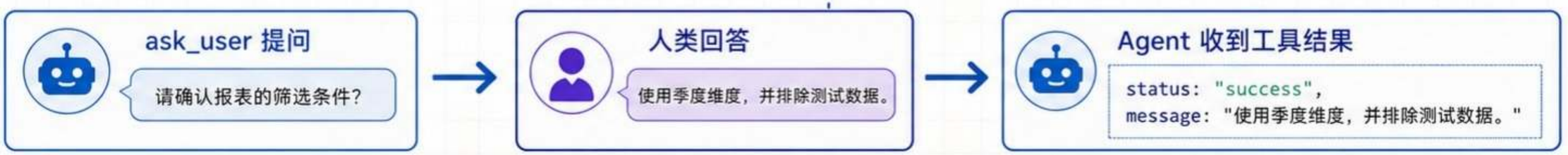
```python
decisions = [{
    "type": "respond",
    "message": "使用季度维度，并排除测试数据。"
}]

agent.invoke(
    Command(resume={"decisions": decisions}),
    config=config,
    version="v2",
)
```

1 工具本身不会执行

2 message 作为成功工具结果

3 只适合 ask_user 这类工具

respond 不是拒绝高风险动作用 reject



批量工具调用：按顺序恢复

decisions 必须和 action_requests 一一对应

action_requests

1. delete_file(
   path="temp.txt")
2. send_email(
   to="admin@example.com")
3. ask_user(
   question="使用哪个周期？")

1 → 1

2 → 2

3 → 3

PYTHON

```python
decisions = [
    {"type": "approve"},
    {"type": "reject", "message": "不要发送邮件。"},
    {"type": "respond", "message": "使用季度。"},
]

agent.invoke(
    Command(resume={"decisions": decisions}),
    config=config,
    version="v2"
)
```

数量相同 | 顺序相同 | 每个动作一个 decision

子 Agent：单独配置中断

主 Agent 可信，不代表子 Agent 可以放开权限

主 Agent

interrupt_on:

delete_file	True
read_file	False

子 Agent file-manager

interrupt_on:

read_file	True
delete_file	True

更严格的审批

PYTHON

```python
subagents = [{
    "name": "file-manager",
    "interrupt_on": {
        "read_file": True,
        "delete_file": True,
    },
}]
```

子 Agent 可覆盖主配置

敏感数据读取也可审批

权限按职责分层

文件系统权限中断

保护敏感路径，也可以放到 agent 配置里

```python
from langgraph.checkpoint.memory import MemorySaver

agent = create_deep_agent(
    model=model,
    permissions=[
        FilesystemPermission(
            operations=["write"],
            paths=["/secrets/**"],
            mode="interrupt",
        )
    ],
    checkpointer=MemorySaver(),
)
```

1. operations：只拦截写入

仅拦截危险的写入操作，读操作不受影响

2. paths：保护 /secrets/**

仅对敏感路径生效，其他路径正常访问

3. mode="interrupt"：暂停并等待确认

触发时中断执行，等待人工确认后再恢复

配置门禁 → 展示中断 → Command(resume=...) 恢复

本讲回顾

把工具审批变成可恢复的 Agent 流程

- `interrupt_on` 决定哪些工具需要人工审查
- `allowed_decisions` 限定 approve / edit / reject / respond
- `thread_id` + `checkpointer` 保存暂停现场
- `action_requests` 与 `decisions` 一一对应
- 子 Agent 可以按职责单独配置中断

人类不是旁路，而是运行时的一部分

下一讲： HITL 进阶 —— interrupt()、 checkpoint 与恢复规则