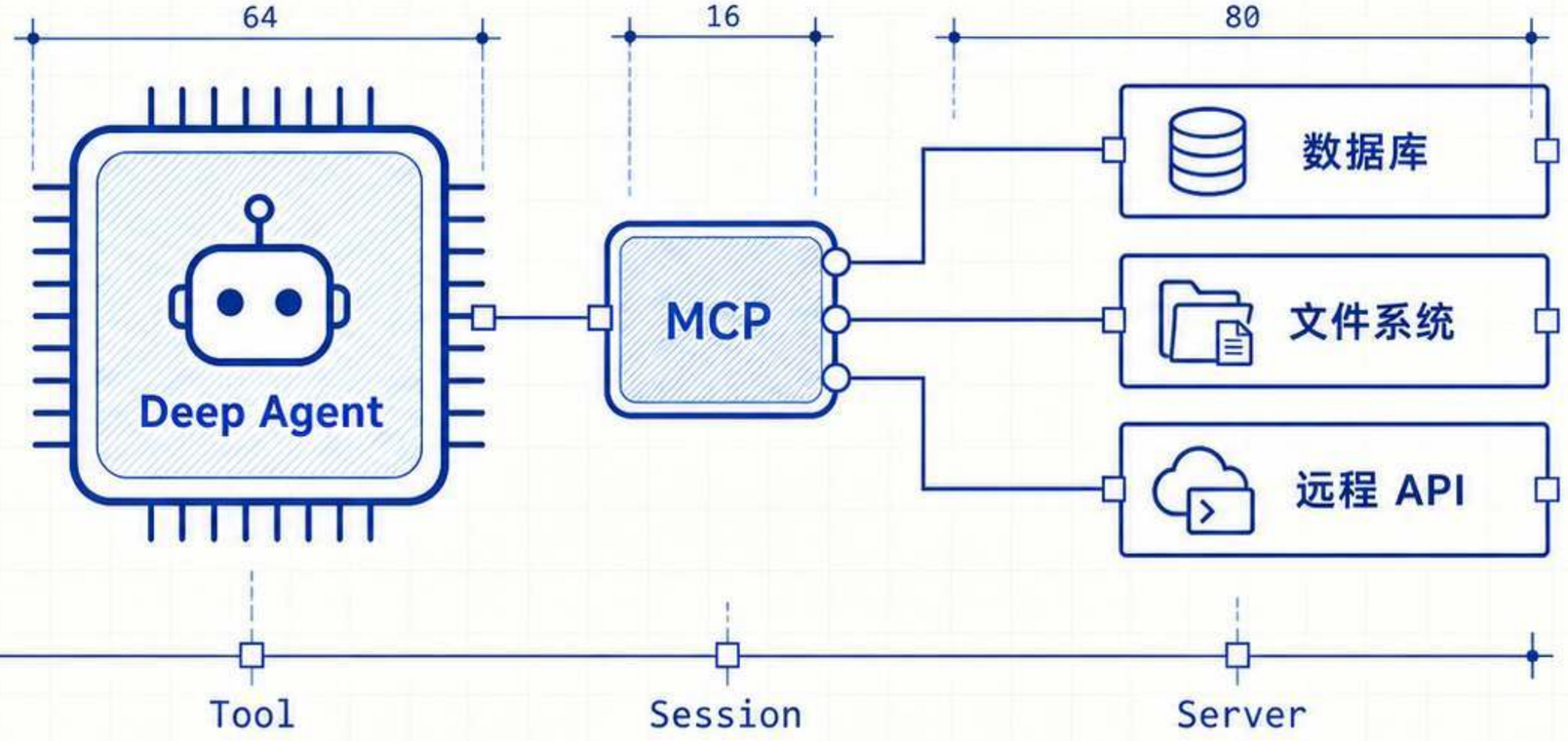
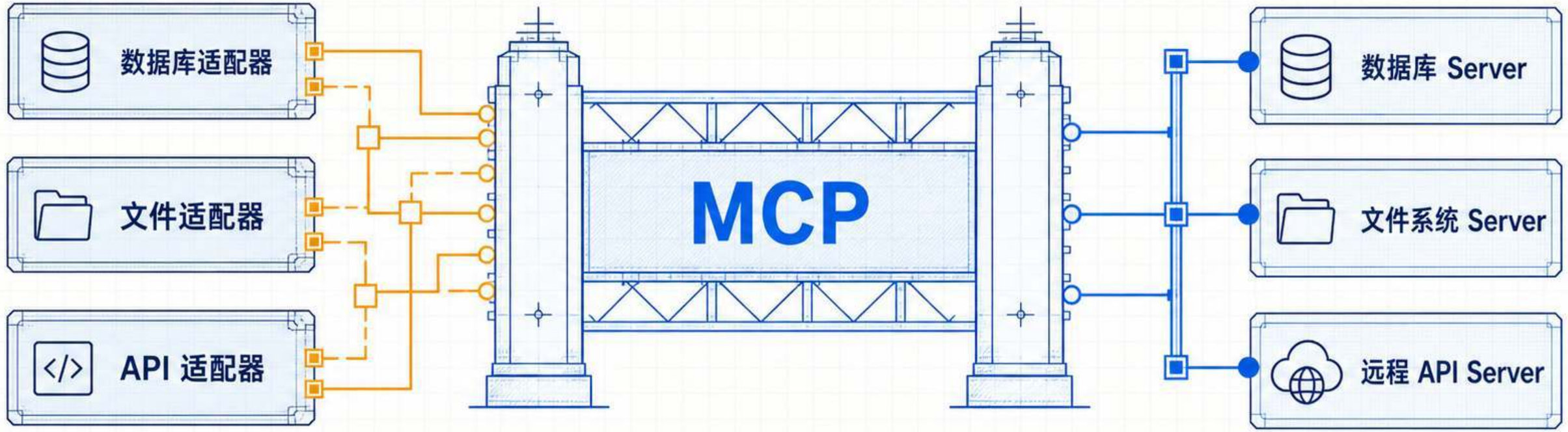
Deep Agents 实战

第 16 讲：MCP — 用标准协议扩展工具生态



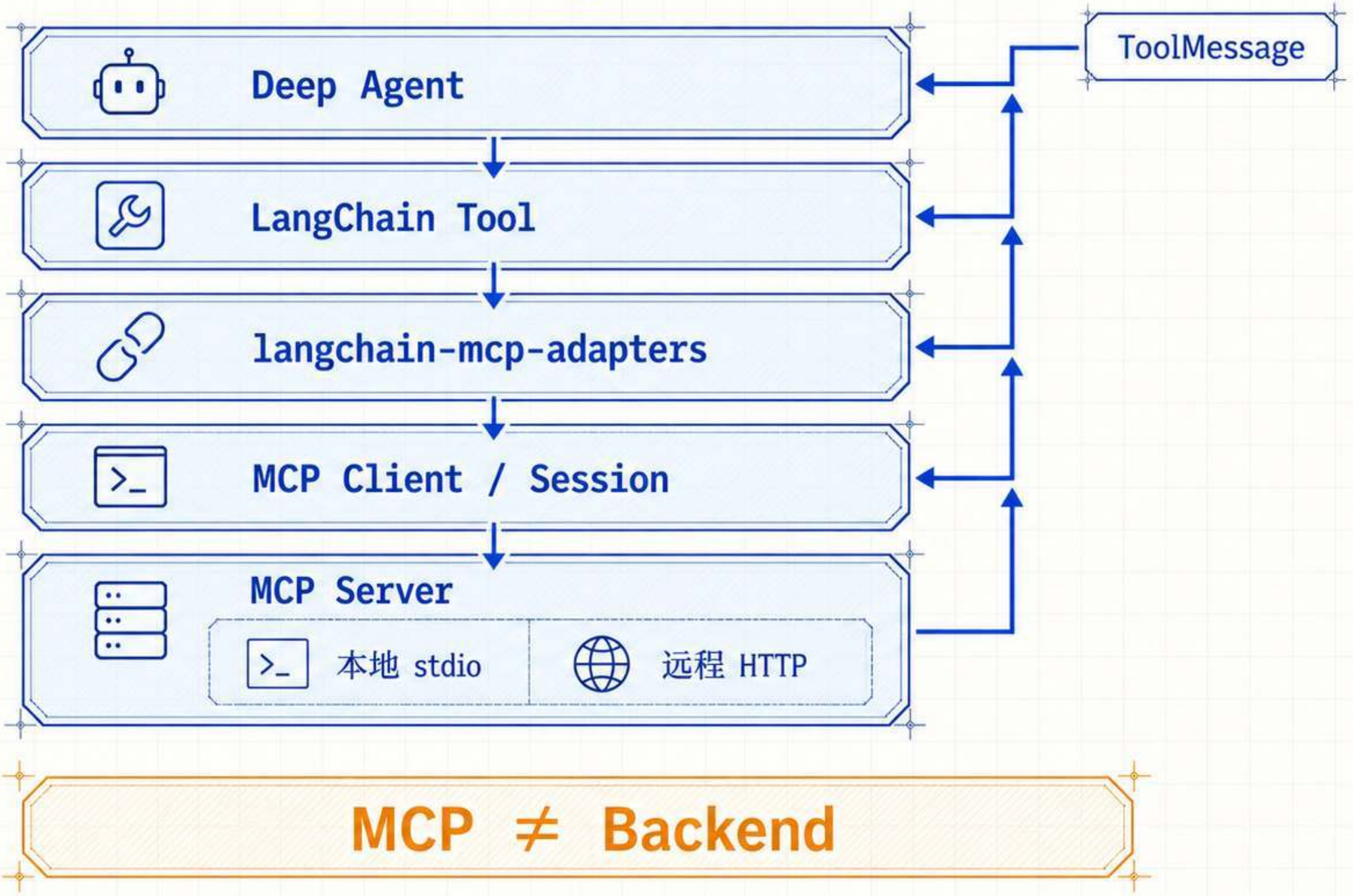
专用适配越多，工具生态越难维护

MCP 把数据库、文件系统与 API 收敛到同一协议边界



MCP 不是 Backend，它位于 Agent 与外部能力之间

五层职责清晰，结果沿反向路径变成 ToolMessage



tools= 只接 Tools，不会自动接入另外两类能力

Resources 与 Prompts 都需要应用主动放入上下文

十几行 FastMCP 代码，就能把函数变成工具

函数名、docstring 与类型标注分别生成 name、description、inputSchema

```python
from mcp.server.fastmcp import FastMCP

mcp = FastMCP("Chapter 12 Math")

@mcp.tool()
def add(a: int, b: int) -> int:
    """Add two integers exactly."""
    return a + b

@mcp.tool()
def multiply(a: int, b: int) -> int:
    """Multiply two integers exactly."""
    return a * b

if __name__ == "__main__":
    mcp.run(transport="stdio")
```

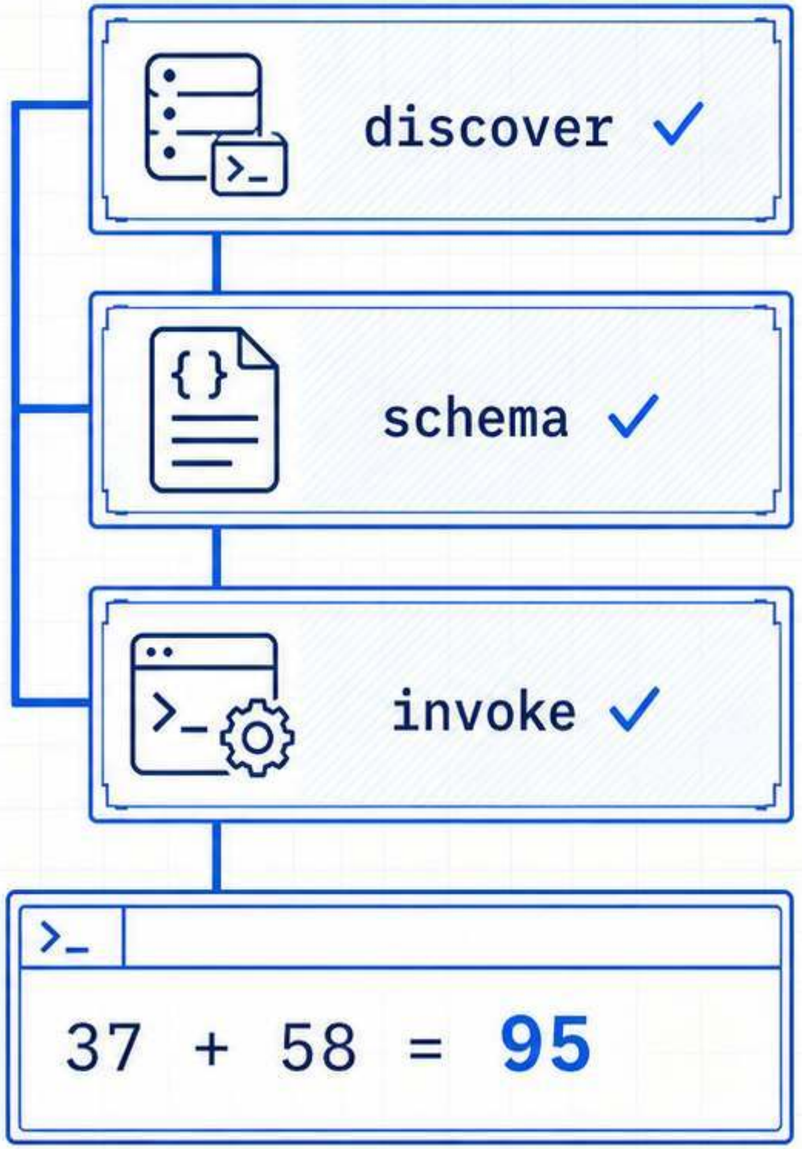
不要先接模型，先逐层证明协议链路

模型与 API Key 暂时都不需要

一个无模型冒烟测试，验证发现、Schema 与真实调用

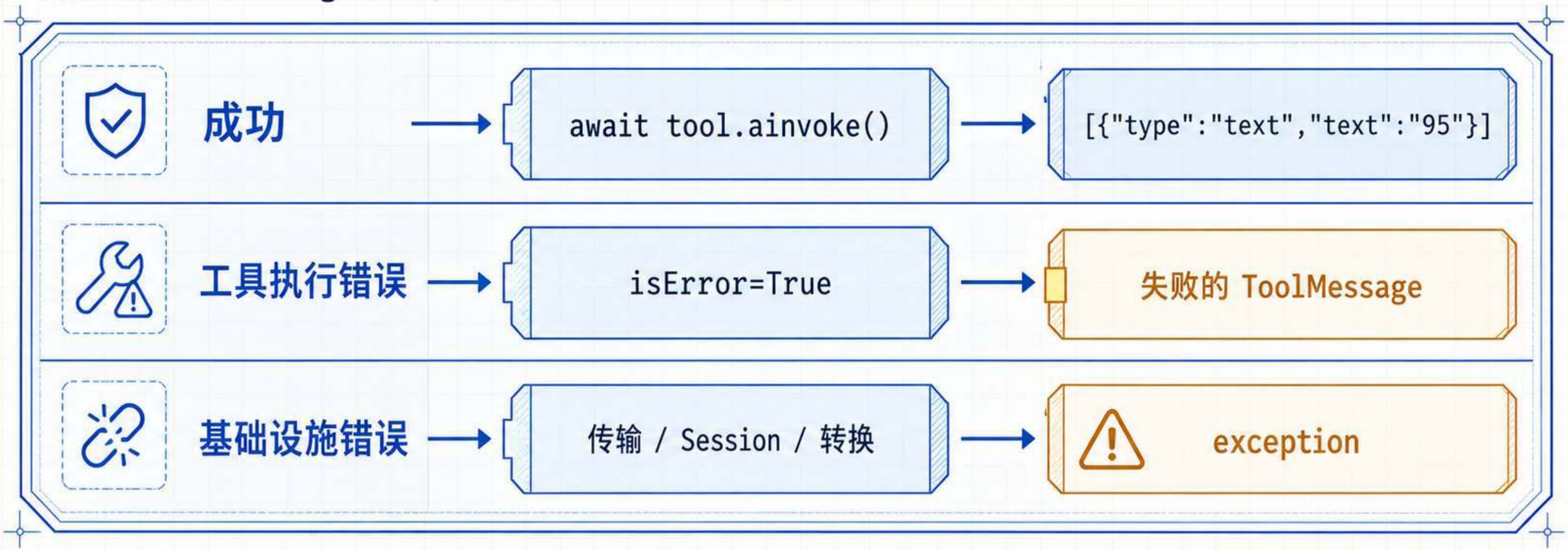
MultiServerMCPClient 只连接本地 course_math

```
SERVER = Path(__file__).with_name("math_server.py").resolve()
client = MultiServerMCPClient(
    {"course_math": {
        "transport": "stdio",
        "command": sys.executable,
        "args": [str(SERVER)],
    }},
    tool_name_prefix=True,
)
tools = await client.get_tools()
add_tool = next(
    t for t in tools
    if t.name == "course_math_add"
)
schema = schema_as_dict(add_tool)
assert schema["required"] == ["a", "b"]
result = await add_tool.ainvoke(
    {"a": 37, "b": 58}
)
```



MCP Tool 是异步工具，成功与失败走不同通道

内容块返回给 LangChain；连接与 Session 错误仍会抛出

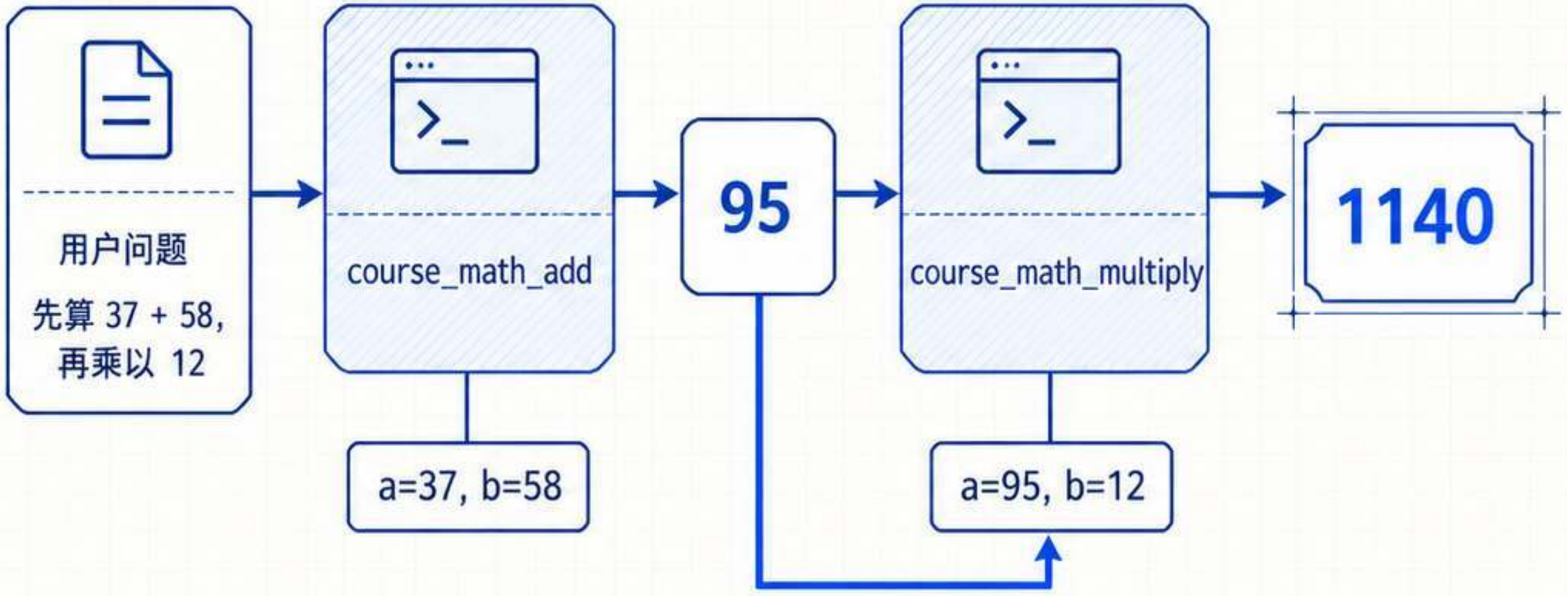


协议通过后，Deep Agent 只需要接收工具列表

正确性应检查工具序列，而不是模型措辞

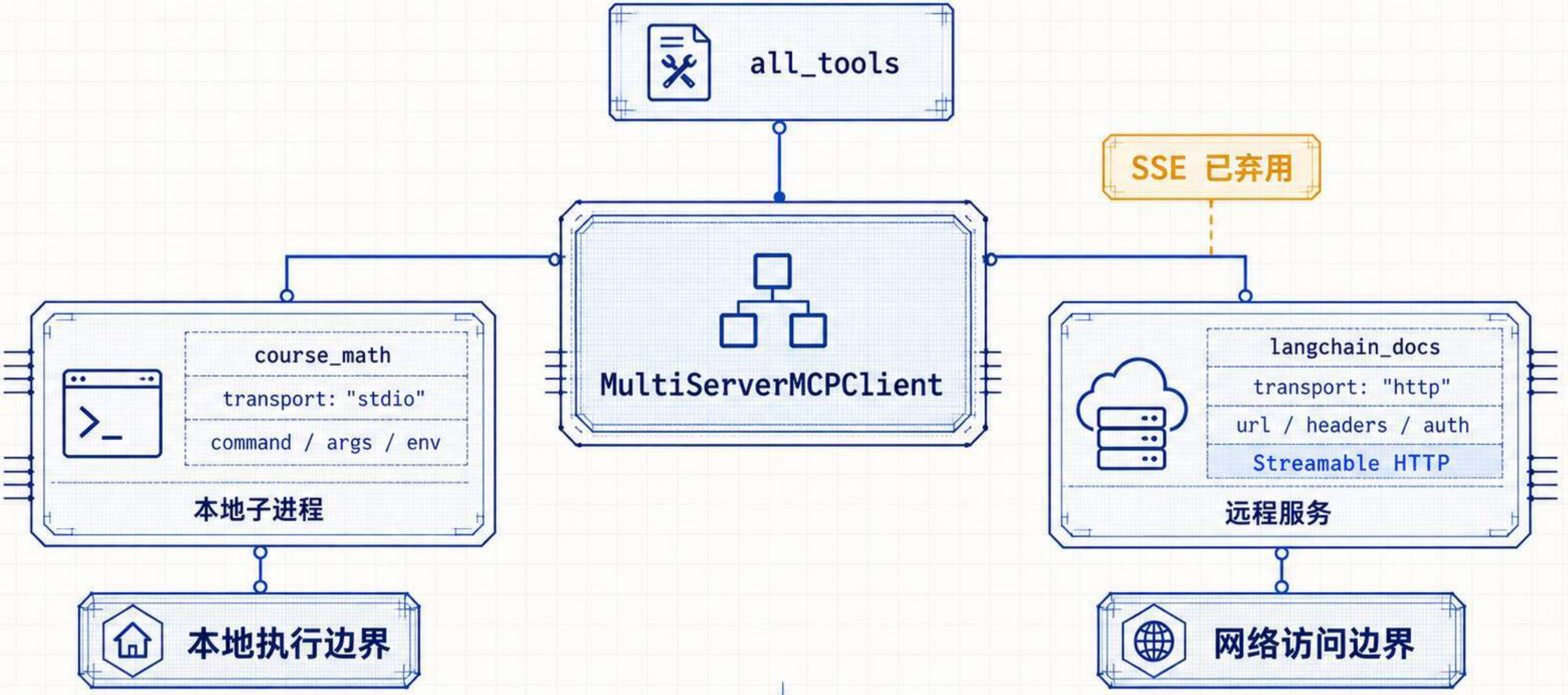
```
tools = await client.get_tools()

agent = create_deep_agent(model=model, tools=tools)
```



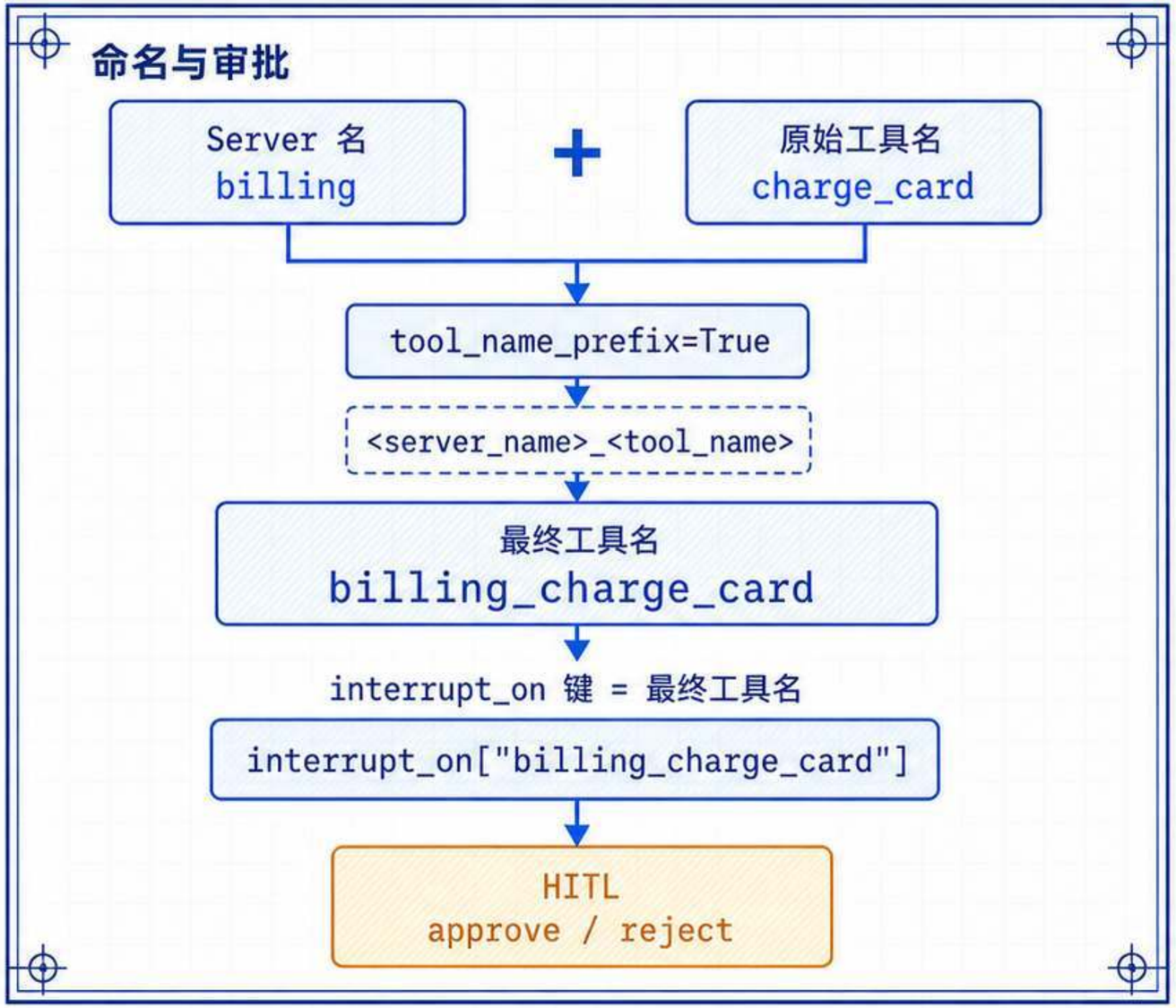
一个 Client 可以同时编排本地与远程 MCP Server

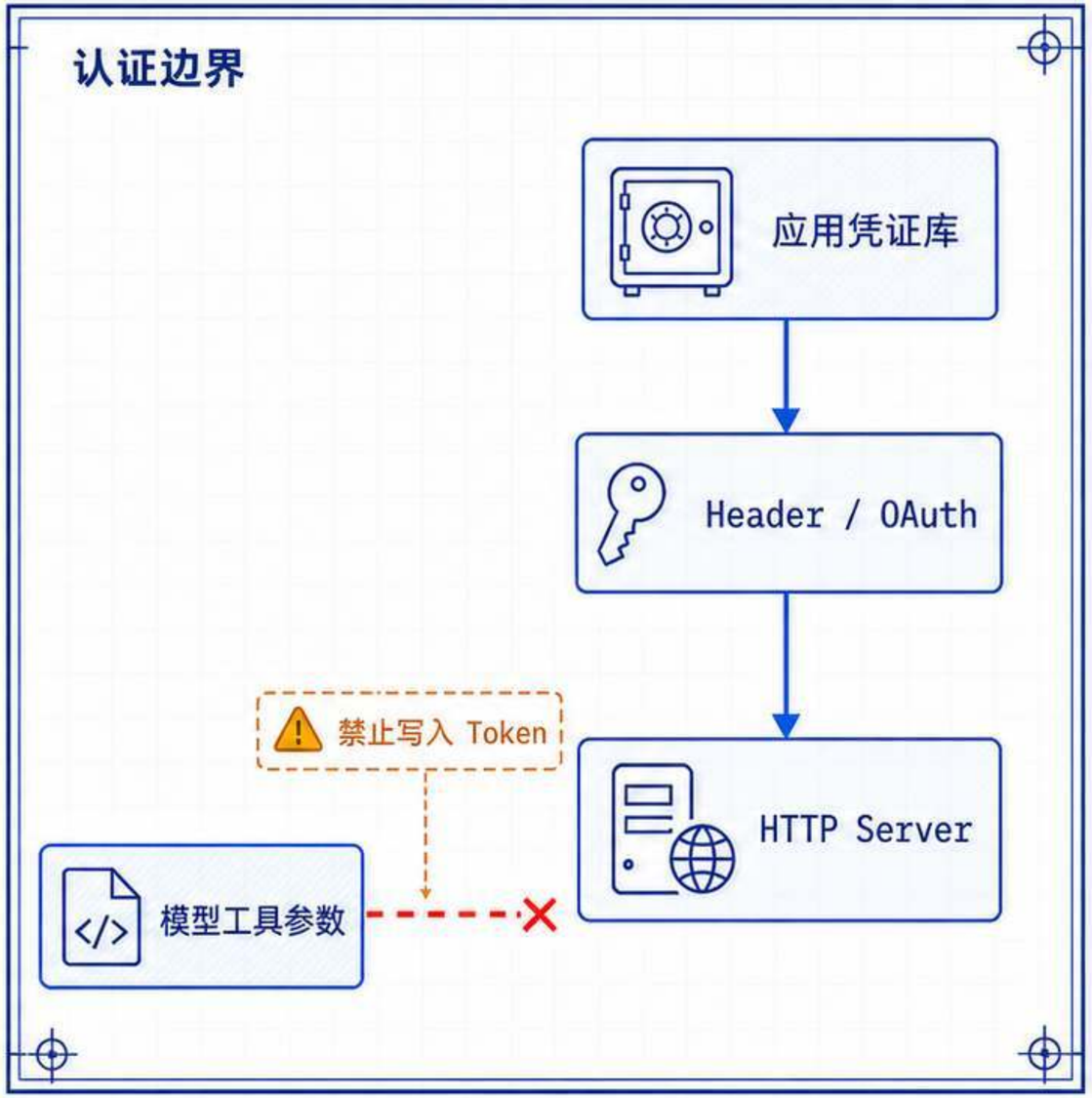
stdio 管本地子进程，Streamable HTTP 管远程服务



工具名前缀让命名、审批与审计保持稳定

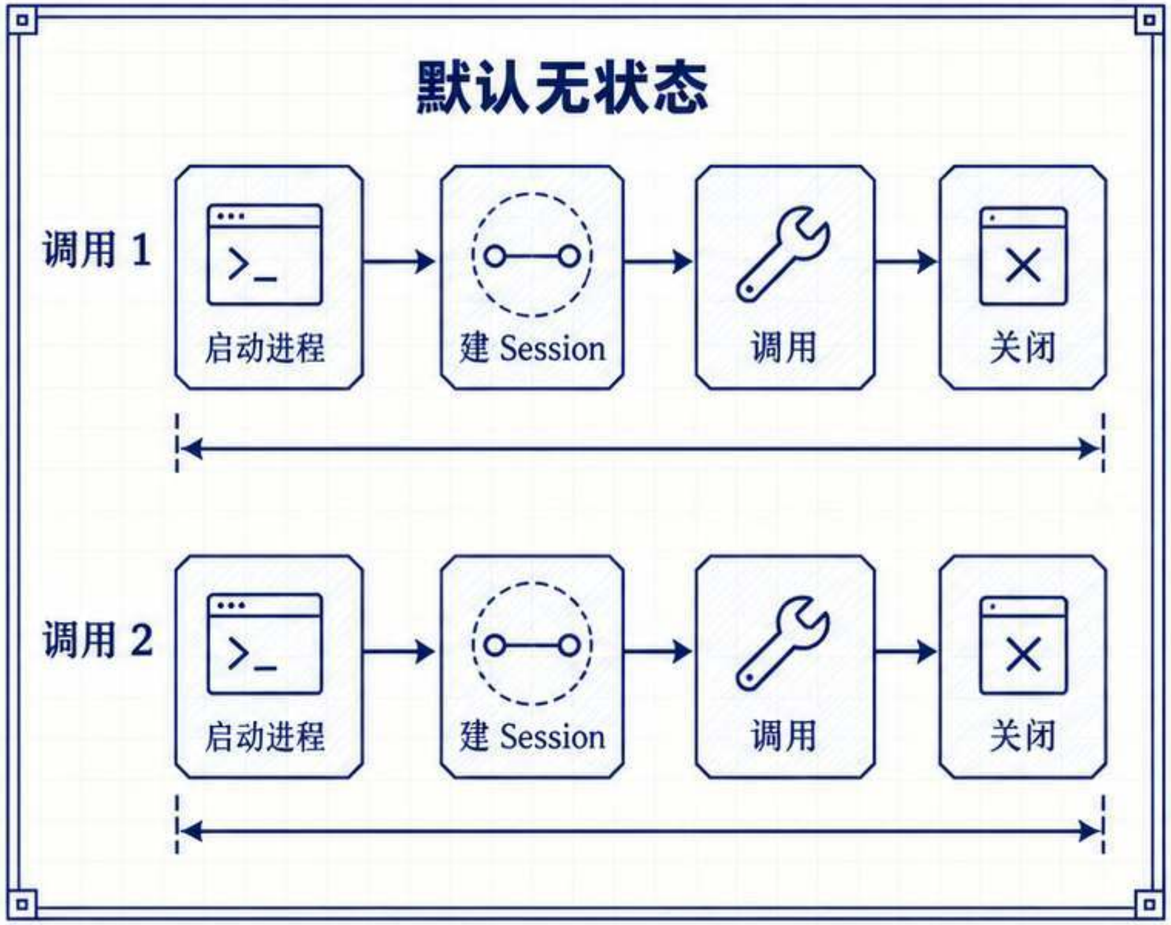
认证由应用注入，不能让模型填写 Token

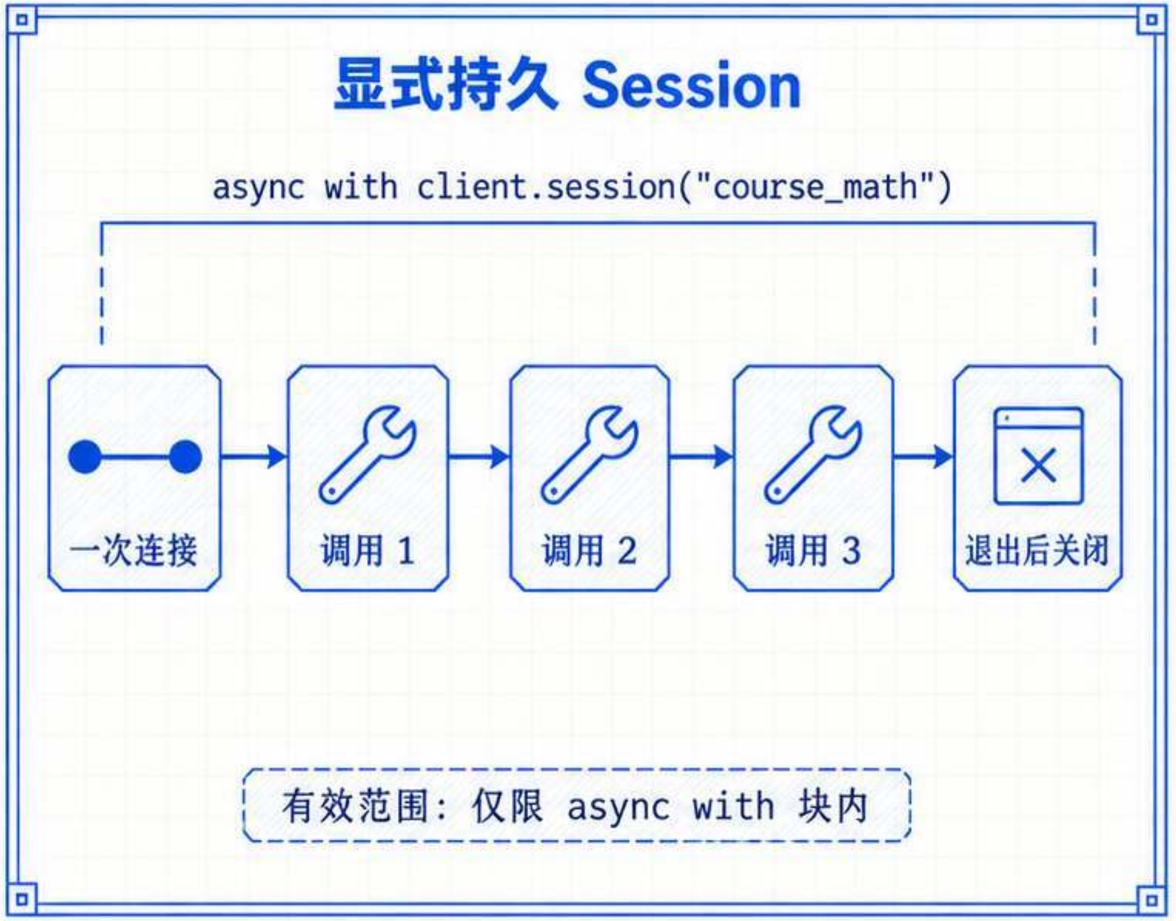




默认每次调用都重建 Session，持久模式才会复用

两种模式的隔离、成本与状态语义不同





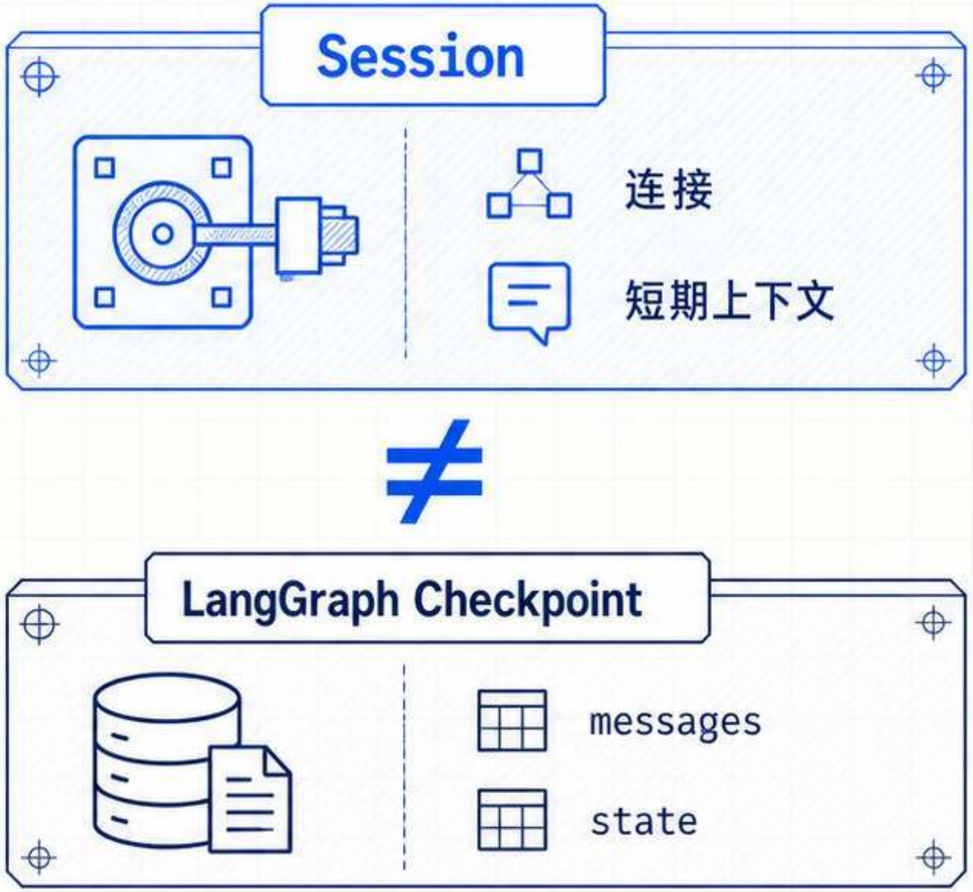
Session 复用连接，Checkpoint 保存 Agent 状态

暂停、重启与 HITL 恢复不能依赖进程内存

持久 Session

```python
async with client.session("course_math") as session:
    tools = await load_mcp_tools(
        session,
        callbacks=client.callbacks,
        tool_interceptors=client.tool_interceptors,
        server_name="course_math",
        tool_name_prefix=client.tool_name_prefix,
        handle_tool_errors=client.handle_tool_errors,
    )
```

生命周期边界

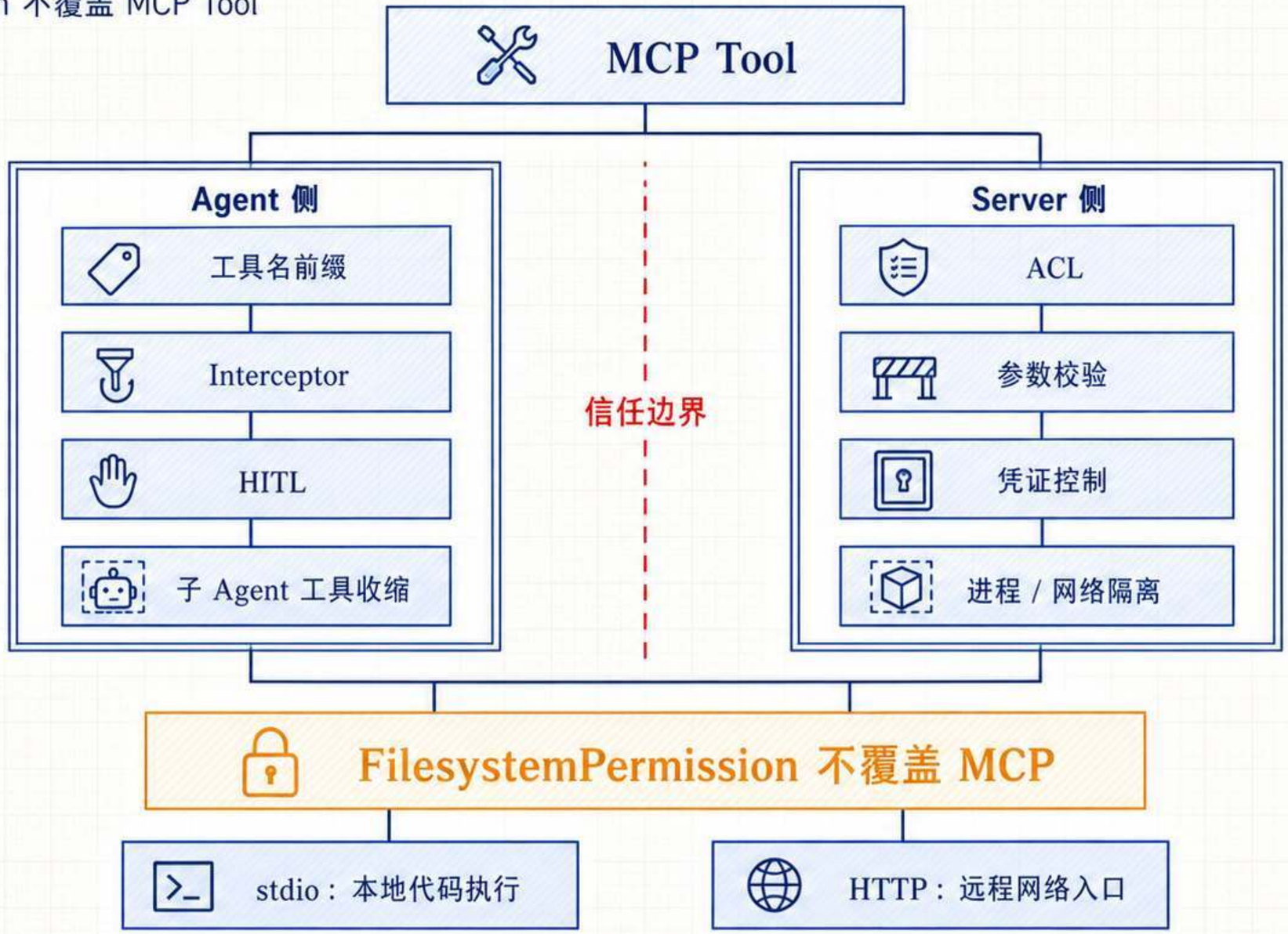


Server DB

长期业务状态 · 幂等写入

MCP 工具要同时守住 Agent 侧与 Server 侧

FilesystemPermission 不覆盖 MCP Tool



AgentSeek 新模板：MCP 从脚手架到运行

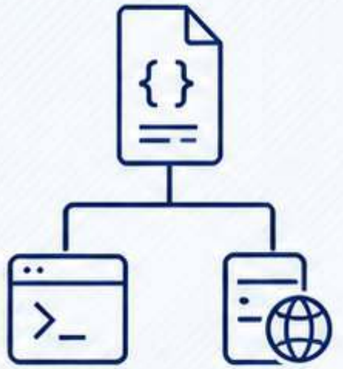
deepagents/mcp 把双传输验证、生命周期与流式 UI 一起打包

安装与创建

```
uv tool install agentseek
```

```
agentseek create deepagents/mcp --no-input
```

.mcp.json



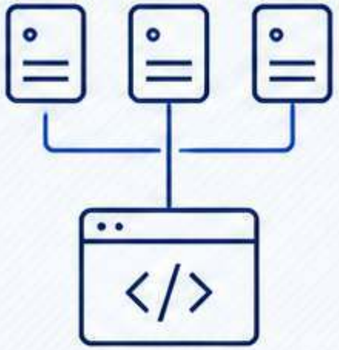
stdio + Streamable HTTP

agentseek task mcp-smoke

无需模型

95 / 2146

agentseek dev



LangGraph + React UI

v1：MCP Tools only

不包含 Resources / Prompts / OAuth / 持久 Session

先证协议链路，再接模型、会话与权限

Server 函数 ⟶ MCP 发现 ⟶ LangChain Tool ⟶ Deep Agent ⟶ 生产控制

无密钥冒烟测试越早变绿，后续问题越容易定位