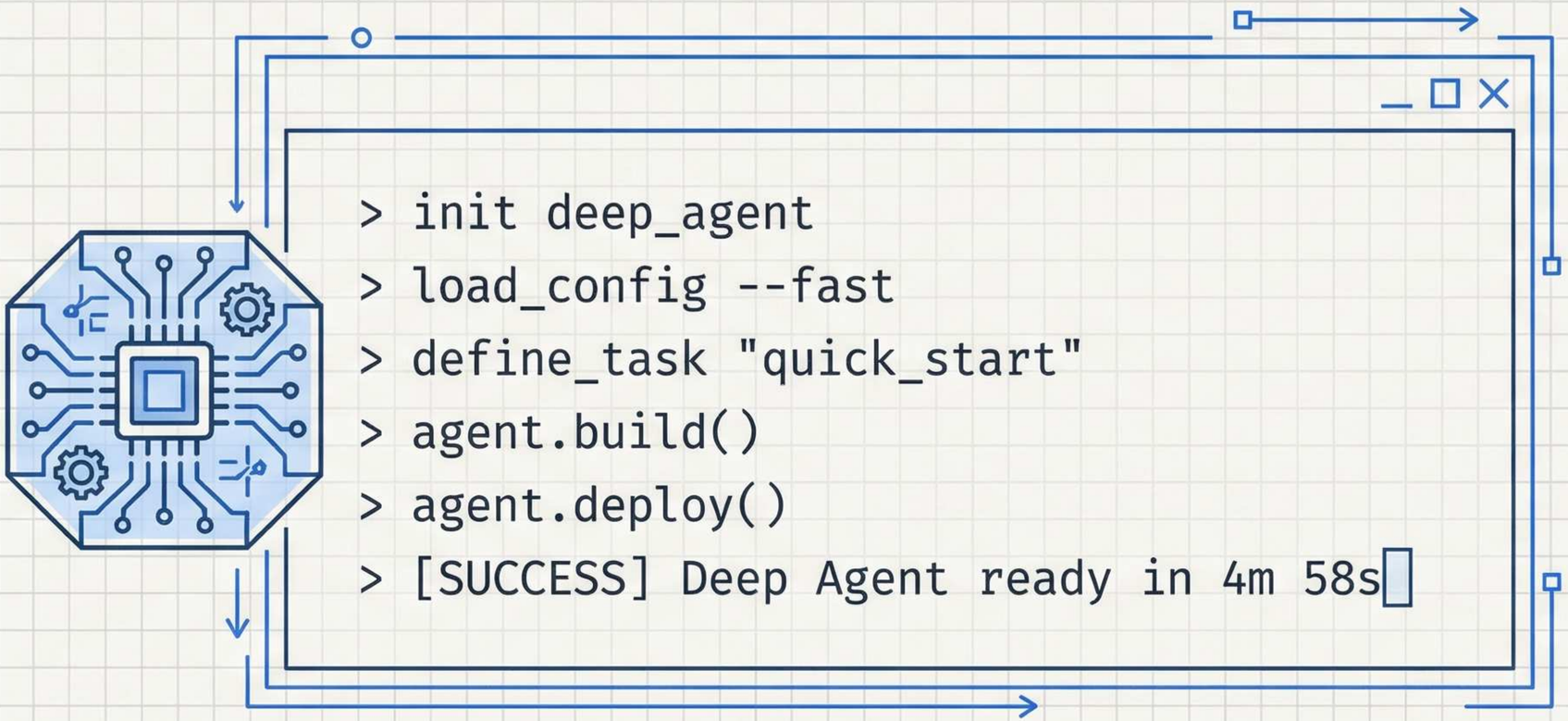
Deep Agents 实战

第 3 讲：快速上手 — 5 分钟构建你的第一个 Deep Agent



安装与配置

两步搞定开发环境

步骤 1 — 安装

```
pip install deepagents langchain-openai
```

Python 3.11+

步骤 2 — 配置 API Key（二选一）

OpenAI 兼容接口

```
export OPENAI_API_KEY=
"your-siliconflow-key"
```

二选一

Anthropic 兼容接口

```
export ANTHROPIC_API_KEY=
"your-siliconflow-key"
```

（Key 值填硅基流动的 API Key）

- 国内直连
- 双接口兼容
  (OpenAI 兼容 / Anthropic 兼容)
- 开源模型丰富

模型接入：三种方式

一张表看懂所有接入方法

方式	代码	适用场景
推荐 ChatOpenAI + base_url（推荐）	`ChatOpenAI(model=..., base_url="https://api.siliconflow.cn/v1")`	第三方平台（硅基流动等）
ChatAnthropic + base_url	`ChatAnthropic(model=..., base_url="https://api.siliconflow.cn")`	Anthropic 兼容平台
字符串格式	`create_deep_agent(model="openai:gpt-4.1")`	官方 API 直连

行业两大标准：OpenAI 兼容（/chat/completions）和 Anthropic 兼容（/messages），国内平台几乎都支持 OpenAI 兼容

推荐模型

免费入门，复杂任务用 SOTA 模型

免费

`Qwen/Qwen2.5-7B-Instruct`

- 7B
- 支持 Tools
- 中文理解优秀
- 轻量快速

推荐模型（复杂任务）

`Pro/zai-org/GLM-5.1` 744B MoE · Agent 任务同类最佳	`Pro/moonshotai/Kimi-K2.6` 1T MoE · 原生多模态，256K 上下文
`Pro/deepseek-ai/DeepSeek-V4-Pro` 671B MoE · 推理和 Agent 能力顶级	`Qwen/Qwen3.6-27B` 27B · 支持思考模式和视觉理解

ⓘ 任务规划、多 Agent 编排等复杂场景，小模型往往无法稳定跑通，请用 SOTA 模型；建议用 MODEL_NAME 环境变量管理模型名

Hello World + 核心参数

create_deep_agent() 三个参数，一个 Agent

```python
from deep_agent import create_deep_agent
from langchain_openai import ChatOpenAI
from my_tools import search_tool

# 模型实例
llm = ChatOpenAI(model="gpt-4")

# 自定义工具列表
custom_tools = [search_tool]

# Agent 角色定义
prompt = "你是一个有用的AI助手。"

# 创建 Agent
agent = create_deep_agent(
    model=llm,
    tools=custom_tools,
    system_prompt=prompt
)

# 执行 Agent
result = agent.invoke({"messages": [{"role": "user", "content": "..."}]})
```

model
模型实例（ChatOpenAI 等）

tools
自定义工具列表

system_prompt
Agent 角色定义

注意：
内置工具自动附带（文件系统、任务规划、子 Agent）

Input: `{"messages": [{"role": "user", "content": "..."}]}`

→ **Agent** (create_deep_agent)

→ **Output:** `result["messages"][-1].content`

工具定义三要素

一个 Python 函数就是一个工具

- 参数类型标注 — 告诉 Agent 每个参数该传什么类型
- Docstring — 告诉 Agent 这个工具的用途（'使用说明书'）
- 默认值 — 标记可选参数，减少 Agent 出错概率

```python
def internet_search(
    query: str,                 # 类型标注
    max_results: int = 5,       # 默认值
) -> dict:
    """Run a web search."""     # Docstring
```

① 参数类型标注与默认值

② 默认值

③ Docstring

要素	对 Agent 的作用
① 参数类型标注	明确输入数据格式，提高调用准确性
② 默认值	简化调用，处理非必要参数，降低错误率
③ Docstring	提供工具用途和使用说明，指导 Agent 正确决策

实战：研究助手

从工具定义到运行结果

1

定义搜索工具

internet_search(query, max_results, topic)
使用 Tavily

2

撰写系统提示词

“你是一位专业的研究员...”

3

创建 Agent

create_deep_agent(model, tools=[internet_search], system_prompt=...)

4

运行

agent.invoke{{"messages": [...]})

运行结果

根据搜索结果，人工智能在医疗领域的最新应用包括诊断辅助、药物研发和个性化治疗。最近的研究重点是深度学习模型如何提高图像分析的准确性，以及生成式AI在加速新药筛选中的潜力。相关挑战包括数据隐私和算法偏见问题。

agent.invoke() 背后发生了什么?

你写 1 行调用，Agent 自动完成 10+ 次工具调用

规划任务

调用 write_todos，拆解为多个子步骤

搜索信息

调用 internet_search，执行多次网络搜索

管理上下文

调用 write_file，将搜索结果写入虚拟文件系统

委派子任务

调用 task，将复杂任务委派给子 Agent

综合报告

从文件系统读取信息，撰写最终报告

1 行代码 → 10+ 次工具调用

LangSmith 调试追踪

两行配置，看透 Agent 的"思考过程"

```
export LANGSMITH_TRACING=true
export LANGSMITH_API_KEY="your-key"
```

自动上传

零代码改动，自动上传

能看到什么：

- 模型调用
  - 输入：'分析销售数据...'
  - 输出：'我需要使用...'
- 工具调用
  - 工具：pandas_analysis
  - 参数：query='...'
  - 返回值：'DataFrame...'
- 子 Agent 执行
  - 子 Agent：'Data Analyst'
  - 任务：'处理缺失值...'
  - 结果：'已完成...'
- 响应
  - 最终回答：'根据分析，销售...'
  - Token 消耗：512
  - 耗时：1.2s

本讲回顾

- 环境搭建：pip install + 硅基流动 API Key
- 模型接入：ChatOpenAI + base_url（推荐） / ChatAnthropic / 字符串格式
- 核心 API：create_deep_agent(model, tools, system_prompt) + invoke()
- 工具定义：Python 函数 + 类型标注 + docstring = 一个工具
- 调试追踪：LangSmith 两行配置，完整可观测性

→ 下一讲：深入虚拟文件系统 — Context Engineering 的核心实现