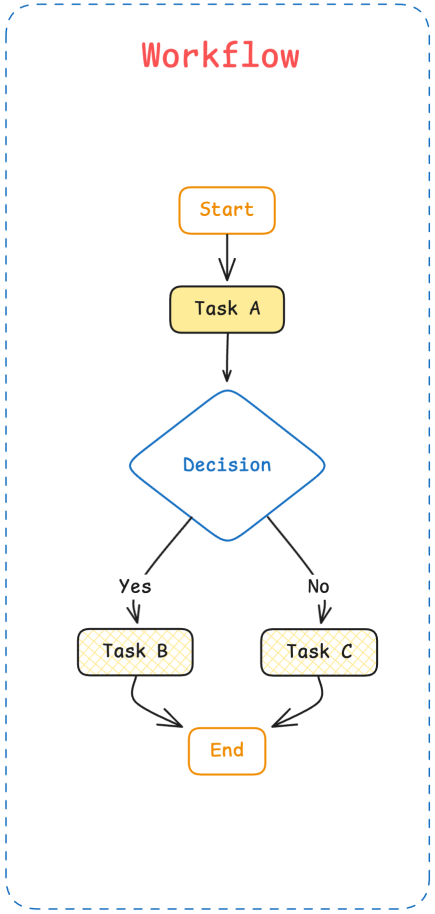
AI大模型应用：NLP与大模型

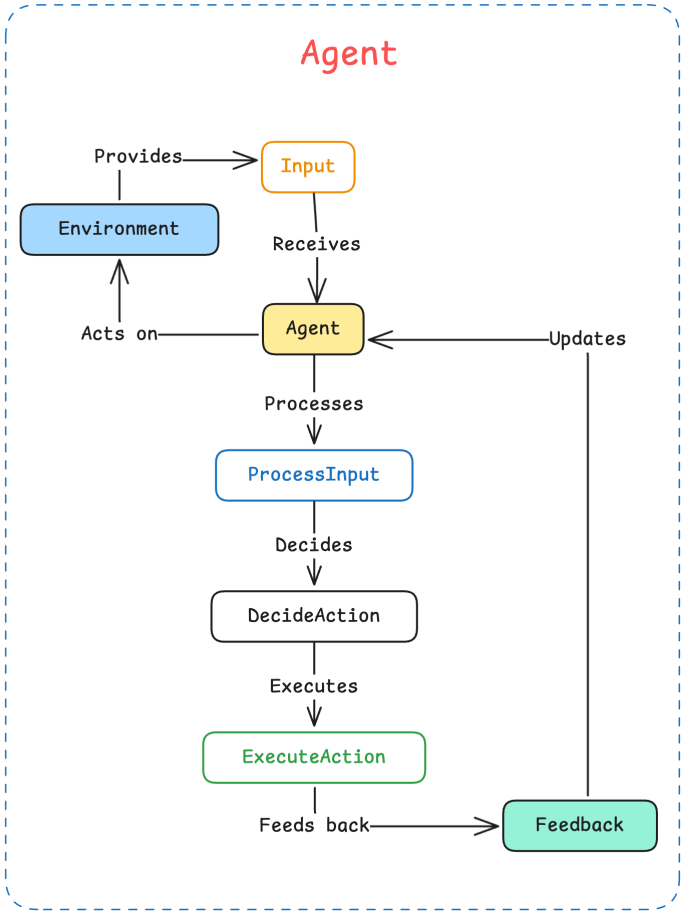
2025年

内部资料，请勿外传

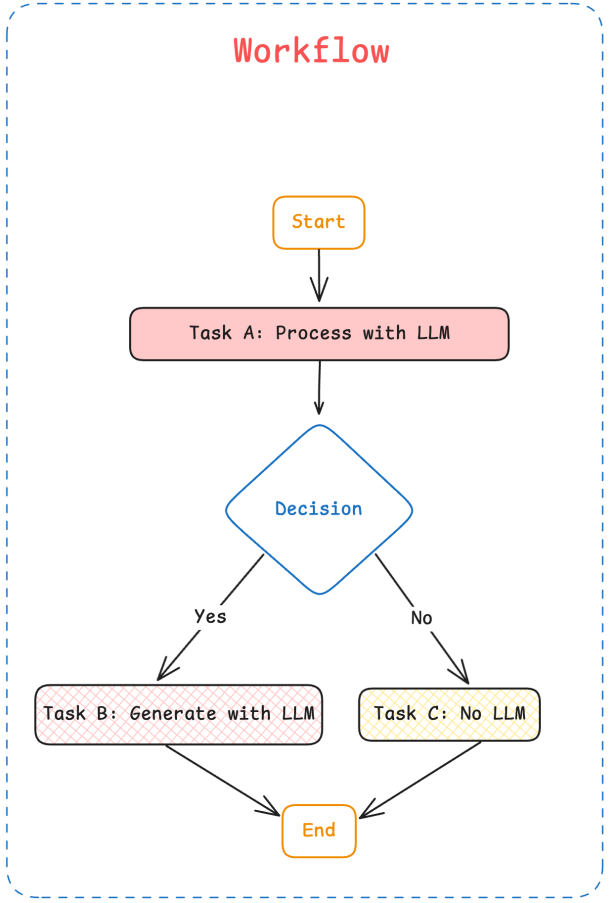
Agent与Workflow

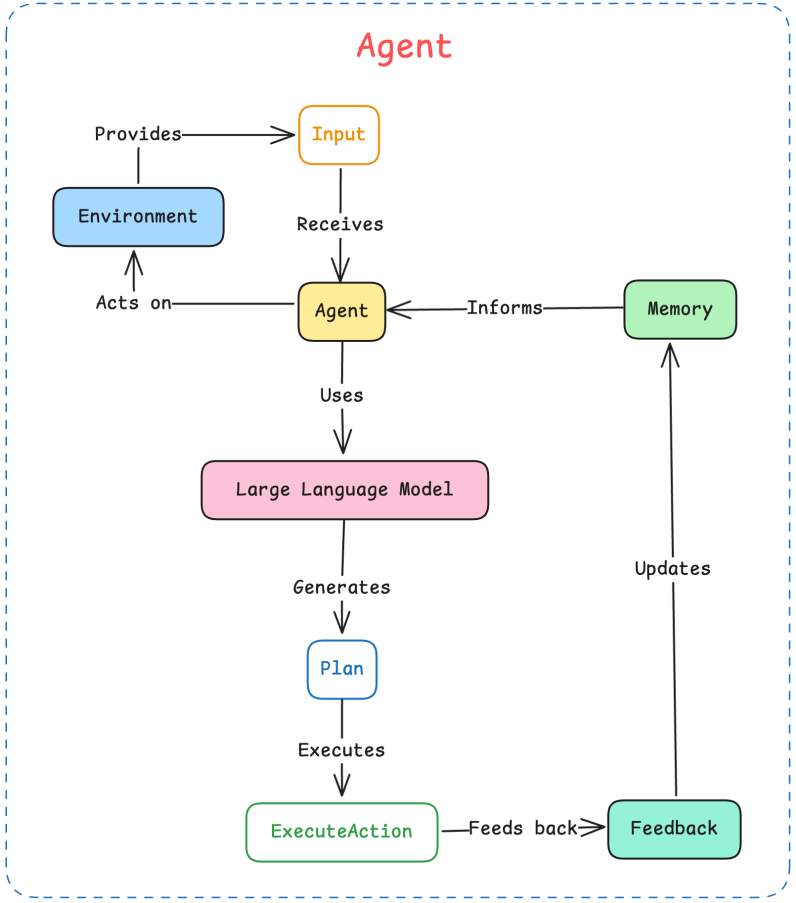
Classic Pattern





Work With Language Models



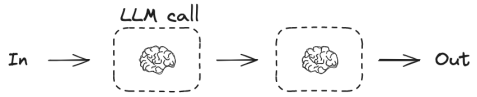


https://huggingface.co/blog/VirtualOasis/agents-vs-workflows-en

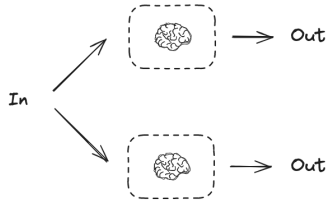
Agent与Workflow

Workflows

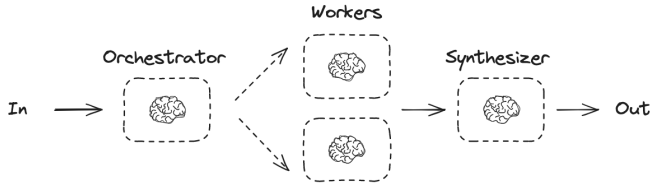
Prompt Chaining



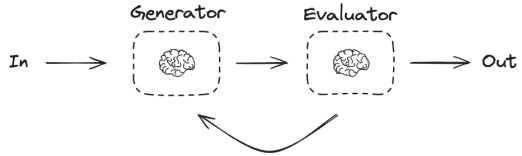
Parallelization



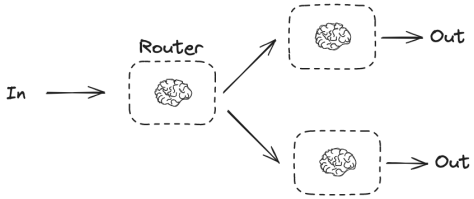
Orchestrator-Worker



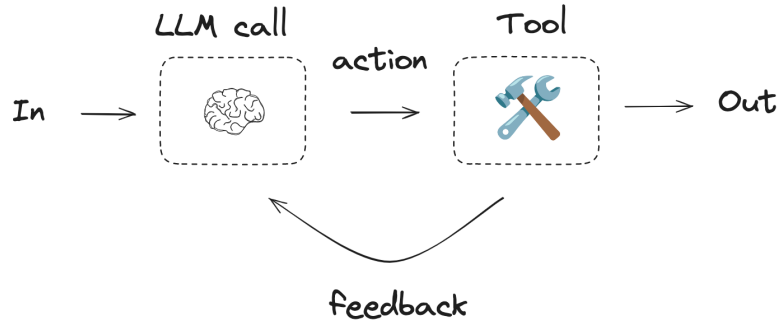
Evaluator-optimizer



Routing



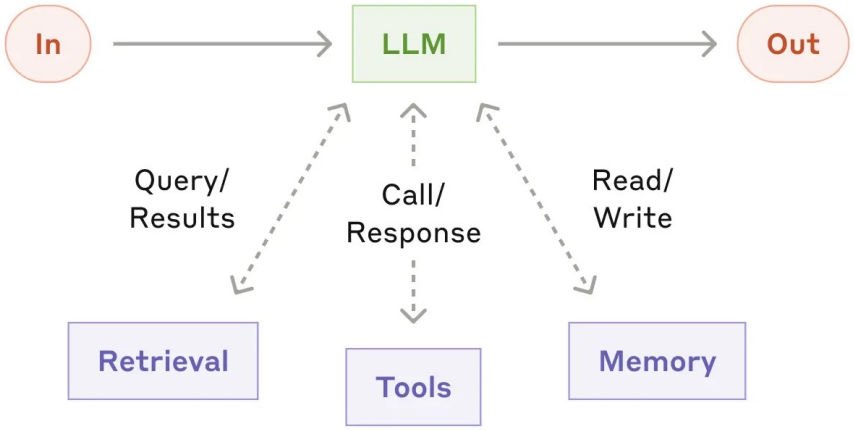
Agent

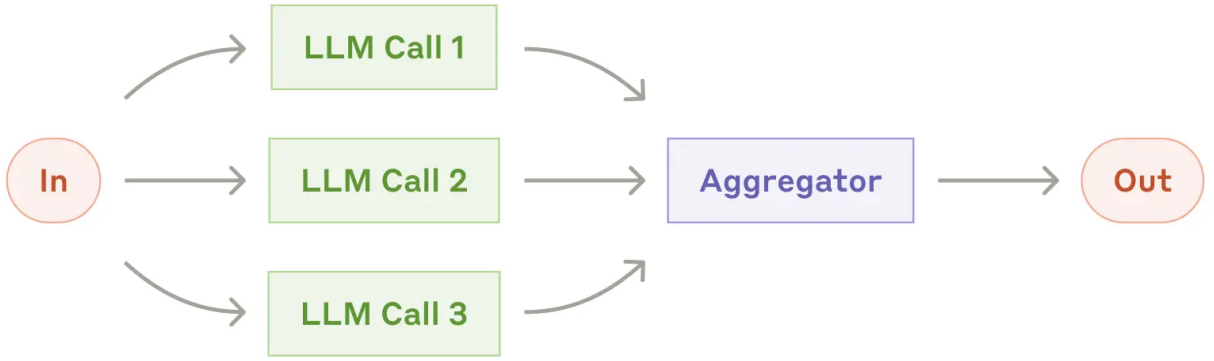


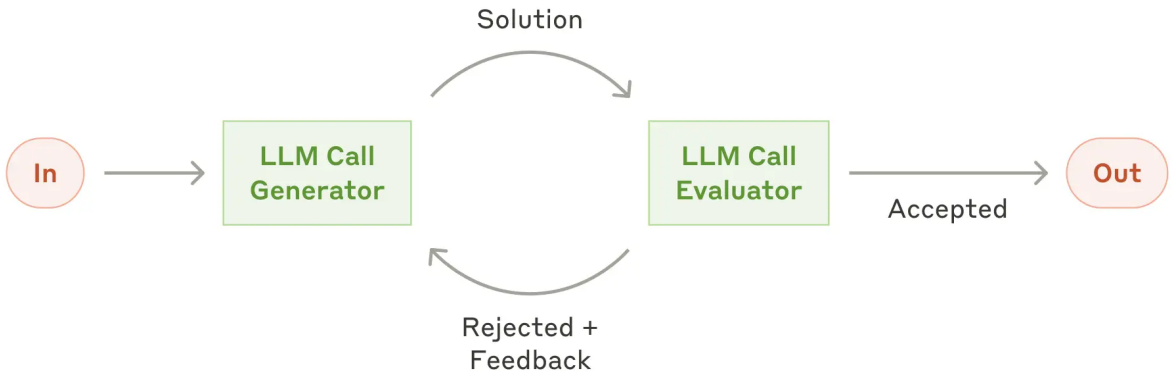
LLM is embedded in predefined code paths	LLM directs control flow through predefined code paths	LLM directs its own actions based on environmental feedback

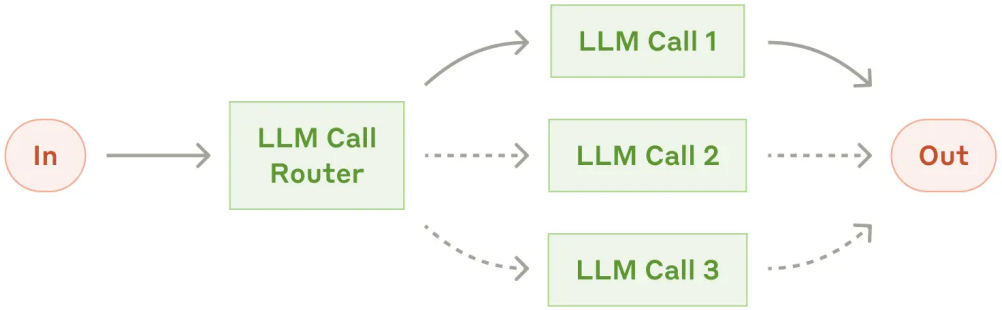
https://docs.langchain.com/oss/python/langgraph/workflows-agents

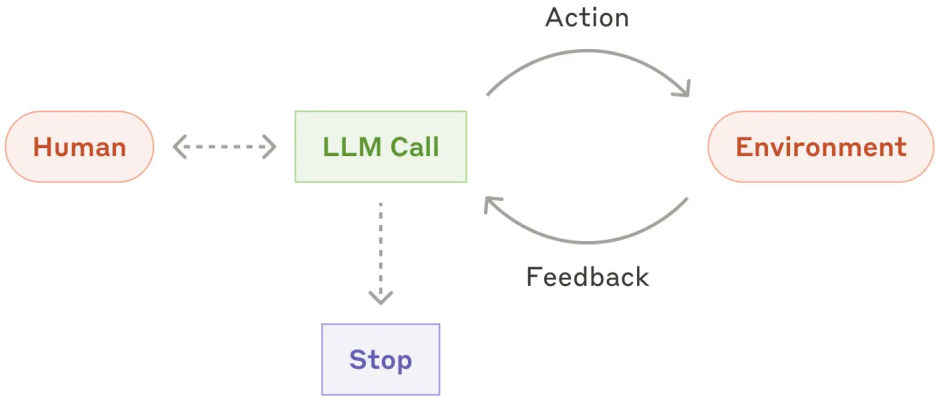
Agent与Workflow

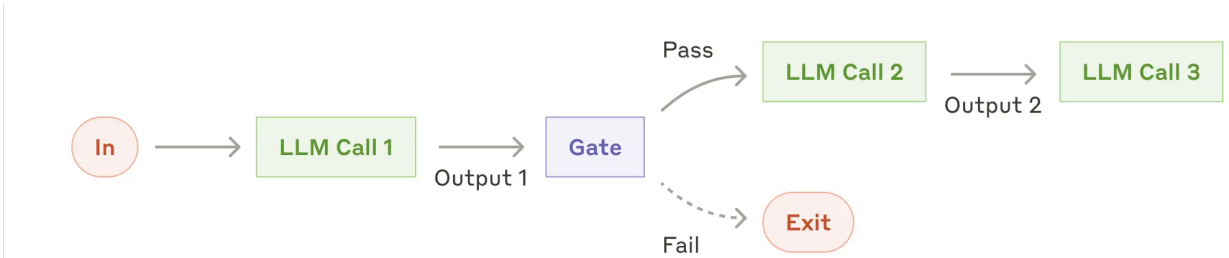












https://www.anthropic.com/engineering/building-effective-agents

Agent与Workflow

特性	智能体 (Agents)	工作流 (Workflows)
核心本质	自主、动态决策的系统。	结构化、按步骤执行的过程。
灵活性	**高。** 能适应新的、独特的或不可预测的情况（例如：客户的独特问题）。	**低/刚性。** 遵循固定的步骤；适用于变化不大的任务（例如：安排维护）。
控制/管理	因其自主性而**较难管理**；动态性可能导致不可靠、错误或无限循环。	每一步都已规划好，因此**更容易控制。**
决策方式	**动态。** LLM（大型语言模型）动态指导自身流程，选择工具，并掌控任务的完成方式（例如：分析问题并决定调取数据库信息）。	**预定义/确定性。** 通过固定的代码路径和规则逻辑进行编排（例如：经理确认 -> 人事部门批准）。
最佳用例	**开放式场景，** 任务无法完全预定义（例如：实时市场数据分析、复杂的客户支持、项目任务优化）。	**要求一致性和合规性的重复性场景**（例如：自动化休假审批流程、库存管理、电子邮件营销活动）。
实施难度	**因其动态和自主性，构建可靠性具有挑战性。**	**依赖预定义规则，因此实施和维护更简单；易于调试和迭代。**

Agent与Workflow

什么时候用Agent 或 Workflow?

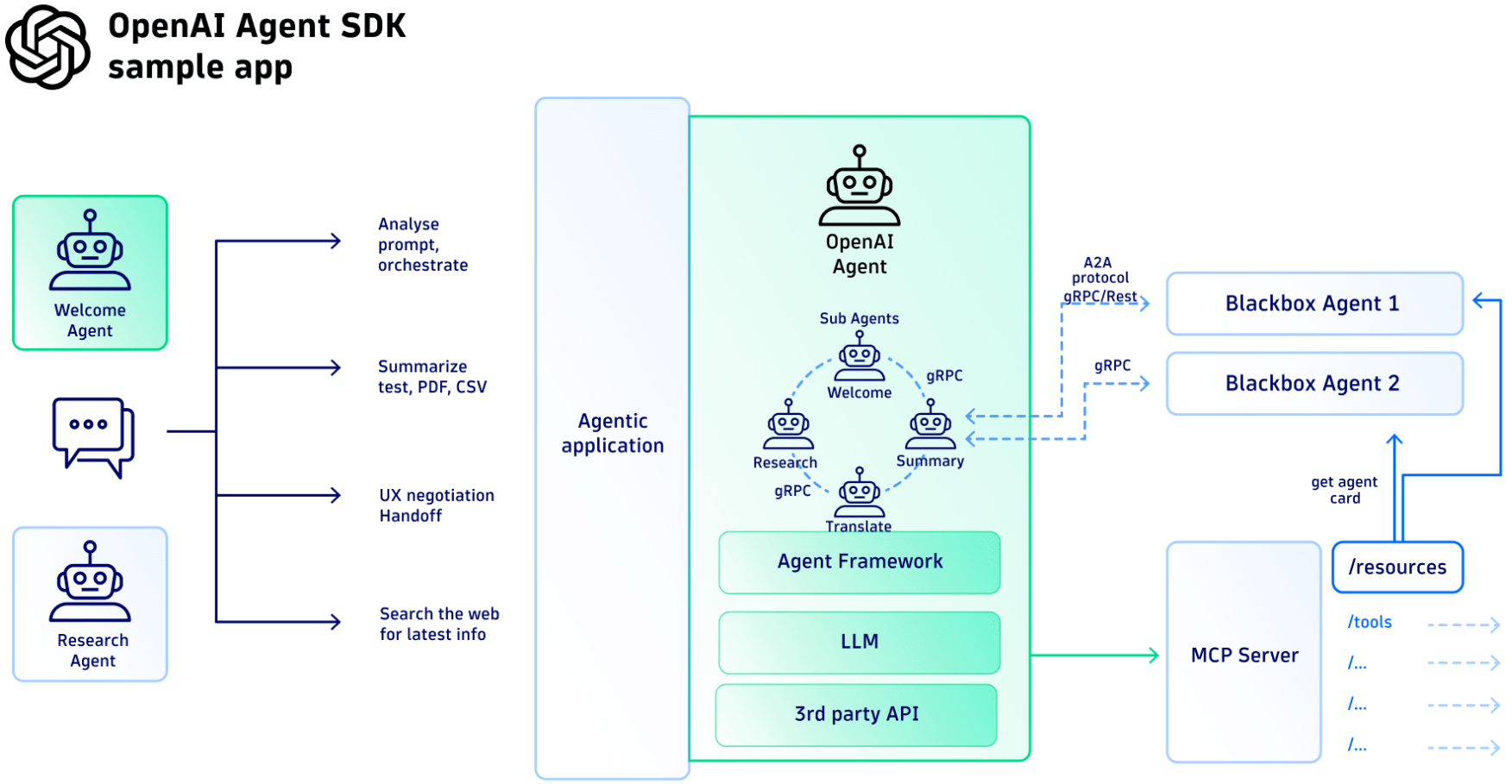
- ✓ Workflow适用场景更加普世，因为90%的应用都是确定流程的。
- ✓ Workflow搭建和调试更加简单一些，并且可见即所得（参考Dify和Coze）。
- ✓ Workflow更多是偏向流程的搭建，其中也可以有Agent的节点。

- ✓ Agent适用需要调用工具 / 需要思考，需要推理 或 需要循环的场景，流程并不固定。
- ✓ Agent拥有不确定性，并且搭建和调试更加复杂，往往都通过代码完成，没有UI界面。
- ✓ Agent没有固定的流程，只需要用户设定目标，然后选择工具 + 逐步完成。

企业内部用Agent还是Workflow?

- ✓ 在类似Dify的平台上搭建Workflow，并且是产品经理/常规后端开发适用的比较多，更加入门。
- ✓ 在开放应用和复杂中Agent更加适合，但需要一定的门槛，能力上限高。

OpenAI Agents框架



https://openai.github.io/openai-agents-python/

OpenAI Agents框架

The OpenAI Agents SDK enables you to build agentic AI apps in a lightweight, easy-to-use package with very few abstractions. The Agents SDK has a very small set of primitives:

- ✓ **Agents**, which are LLMs equipped with instructions and tools
- ✓ **Handoffs**, which allow agents to delegate to other agents for specific tasks
- ✓ **Guardrails**, which enable validation of agent inputs and outputs
- ✓ **Sessions**, which automatically maintains conversation history across agent runs

https://openai.github.io/openai-agents-python/

OpenAI Agents核心概念

Install the Agents SDK

```
pip install openai-agents # or `uv add openai-agents`, etc
```

Set an OpenAI API key

If you don't have one, follow these instructions to create an OpenAI API key.

```
export OPENAI_API_KEY=sk-...
```

- ✓ Python环境
- ✓ 支持国内或国外模型

Agents are defined with instructions, a name, and optional config (such as `model_config`)

```
from agents import Agent

agent = Agent(
    name="Math Tutor",
    instructions="You provide help with math problems. Explain your reasoning at each st
)
```

- ✓ 一个Agent节点是一次大模型调用

https://openai.github.io/openai-agents-python/quickstart/

OpenAI Agents核心概念

Add a few more agents

Additional agents can be defined in the same way. `handoff_descriptions` provide additional context for determining handoff routing

```
from agents import Agent

history_tutor_agent = Agent(
    name="History Tutor",
    handoff_description="Specialist agent for historical questions",
    instructions="You provide assistance with historical queries. Explain important even
)

math_tutor_agent = Agent(
    name="Math Tutor",
    handoff_description="Specialist agent for math questions",
    instructions="You provide help with math problems. Explain your reasoning at each st
)
```

Define your handoffs

On each agent, you can define an inventory of outgoing handoff options that the agent can choose from to decide how to make progress on their task.

```
triage_agent = Agent(
    name="Triage Agent",
    instructions="You determine which agent to use based on the user's homework question
    handoffs=[history_tutor_agent, math_tutor_agent]
)
```

https://openai.github.io/openai-agents-python/quickstart/

- ✓ **Handoffs（切换）**：允许一个代理将任务或请求路由或“切换”给另一个更专业的代理来处理。
- ✓ **Triage Agent（分流代理）**：一个特殊的代理，用于根据输入内容判断并决定将任务切换给哪个专业代理。

OpenAI Agents核心概念

Add a guardrail

You can define custom guardrails to run on the input or output.

```python
from agents import GuardrailFunctionOutput, Agent, Runner
from pydantic import BaseModel


class HomeworkOutput(BaseModel):
    is_homework: bool
    reasoning: str

guardrail_agent = Agent(
    name="Guardrail check",
    instructions="Check if the user is asking about homework.",
    output_type=HomeworkOutput,
)

async def homework_guardrail(ctx, agent, input_data):
    result = await Runner.run(guardrail_agent, input_data, context=ctx.context)
    final_output = result.final_output_as(HomeworkOutput)
    return GuardrailFunctionOutput(
        output_info=final_output,
        tripwire_triggered=not final_output.is_homework,
    )
```

Guardrails（保护机制）：自定义函数，用于在输入或输出阶段添加逻辑检查。

- ✓ **Input Guardrail（输入保护机制）**：在代理开始处理输入之前运行，例如示例中检查问题是否为家庭作业。
- ✓ **Tripwire Triggered（触发警告）**：当保护机制的检查失败时，会抛出异常，阻止代理继续运行。

https://openai.github.io/openai-agents-python/quickstart/

OpenAI Agents官方文档

Agent 使用：https://openai.github.io/openai-agents-python/agents/

Agent 运行：https://openai.github.io/openai-agents-python/running_agents/

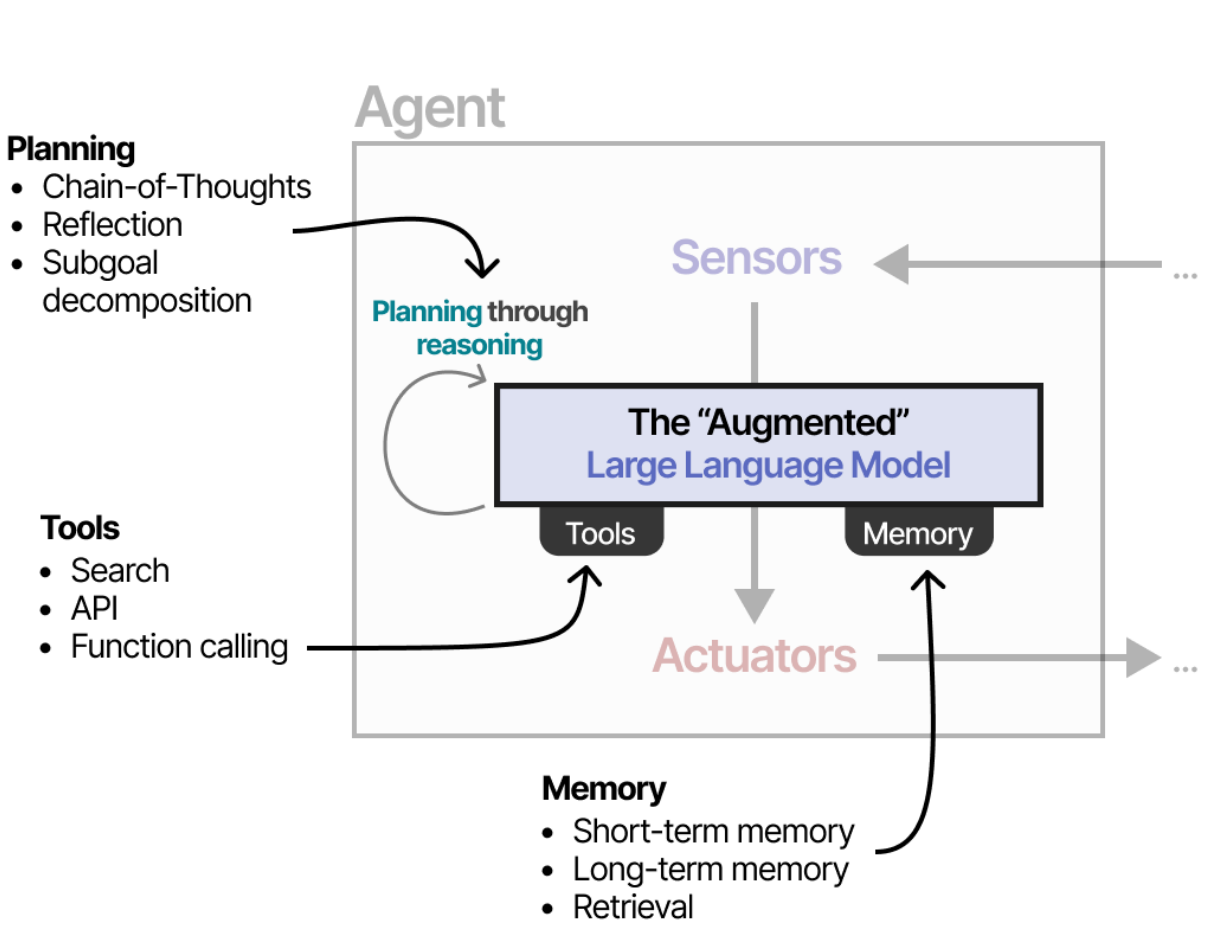
对话缓存：https://openai.github.io/openai-agents-python/sessions/

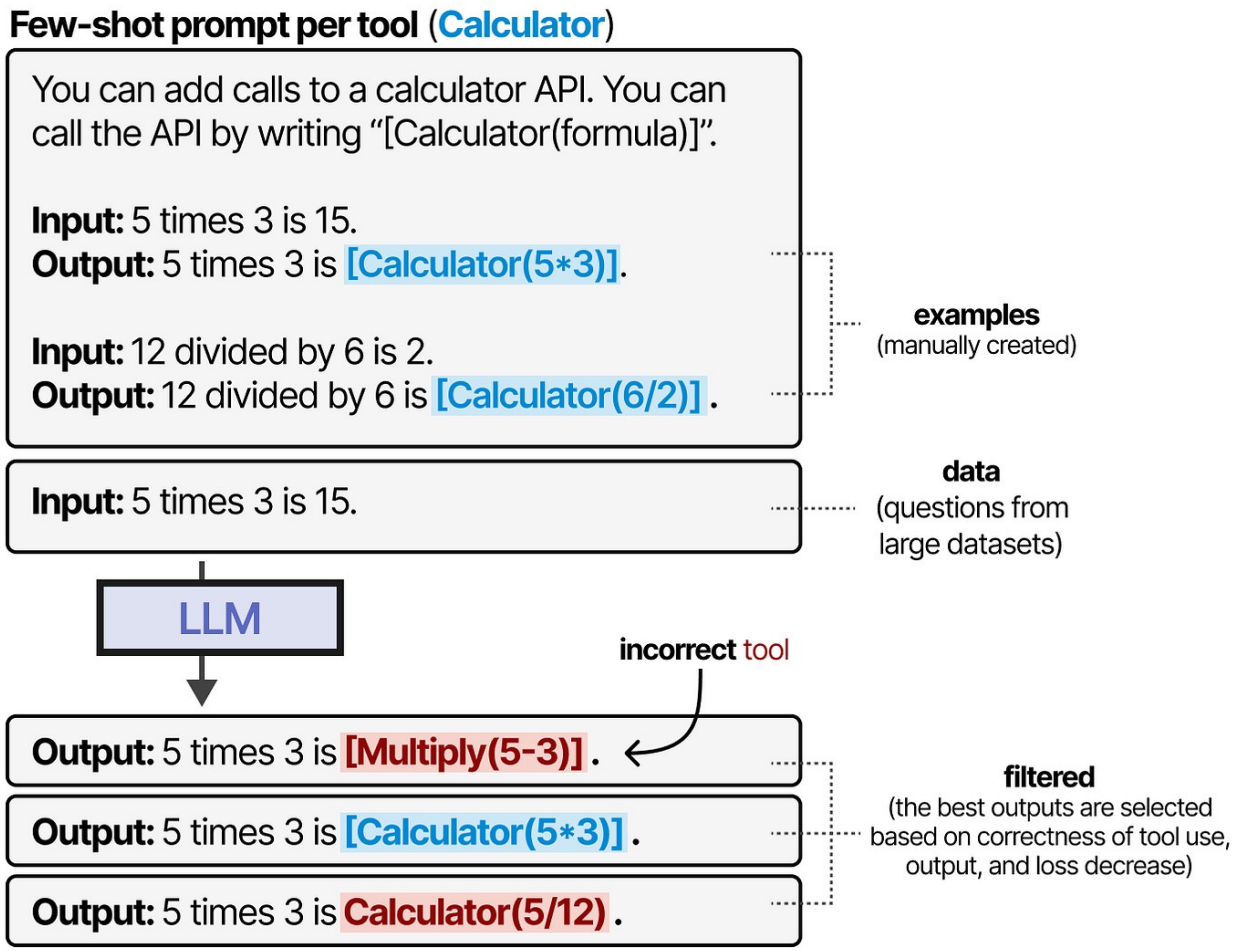
内置工具：https://openai.github.io/openai-agents-python/tools/

MCP支持：https://openai.github.io/openai-agents-python/mcp/

官方案例：https://github.com/openai/openai-agents-python

Agent与Tools





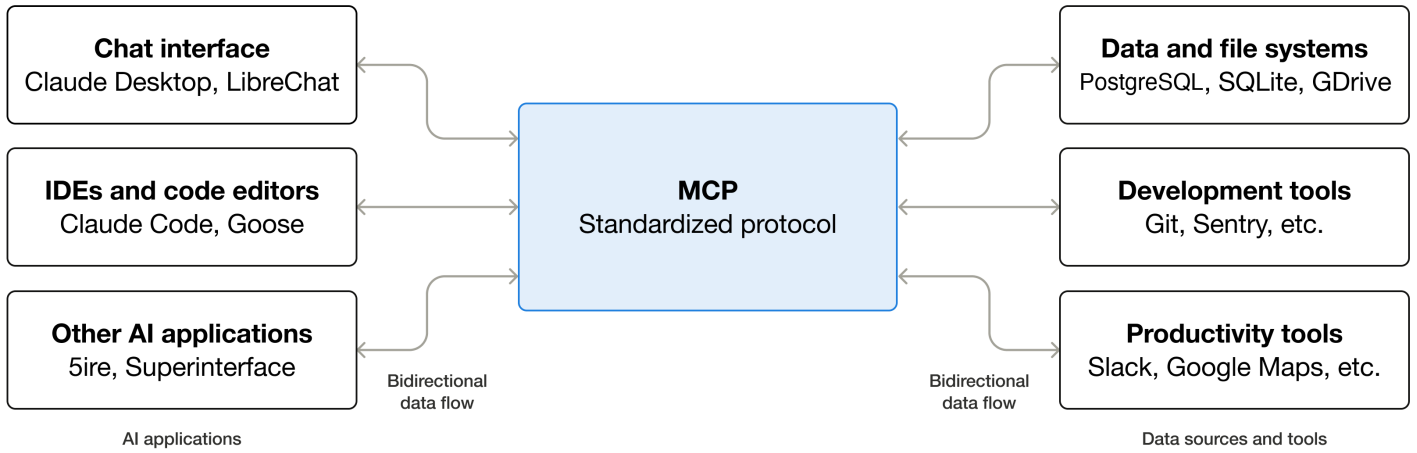
https://newsletter.maartengrootendorst.com/p/a-visual-guide-to-llm-agents

MCP介绍

模型上下文协议 (MCP) 是一个**开源标准**，用于将**AI 应用**（例如 Claude 或 ChatGPT 等大模型）连接到**外部系统。**

MCP 为 AI 应用提供了连接到外部系统的标准化途径，使其能够访问和使用：

- ✓ **数据源:** 例如本地文件、数据库。
- ✓ **工具:** 例如搜索引擎、计算器。
- ✓ **工作流:** 例如专业化的提示（prompts）和流程。

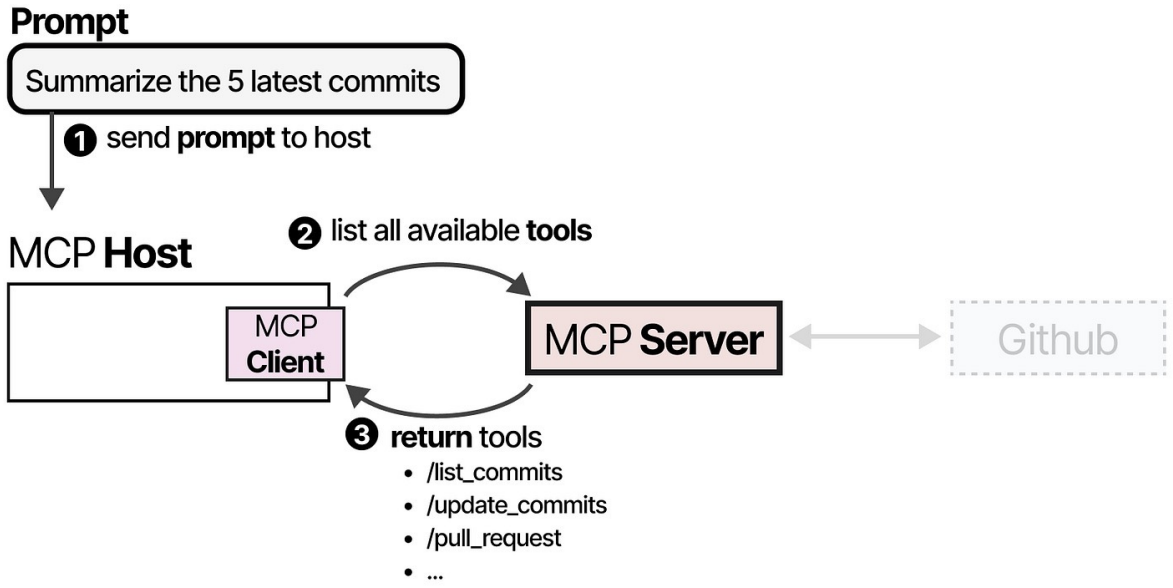


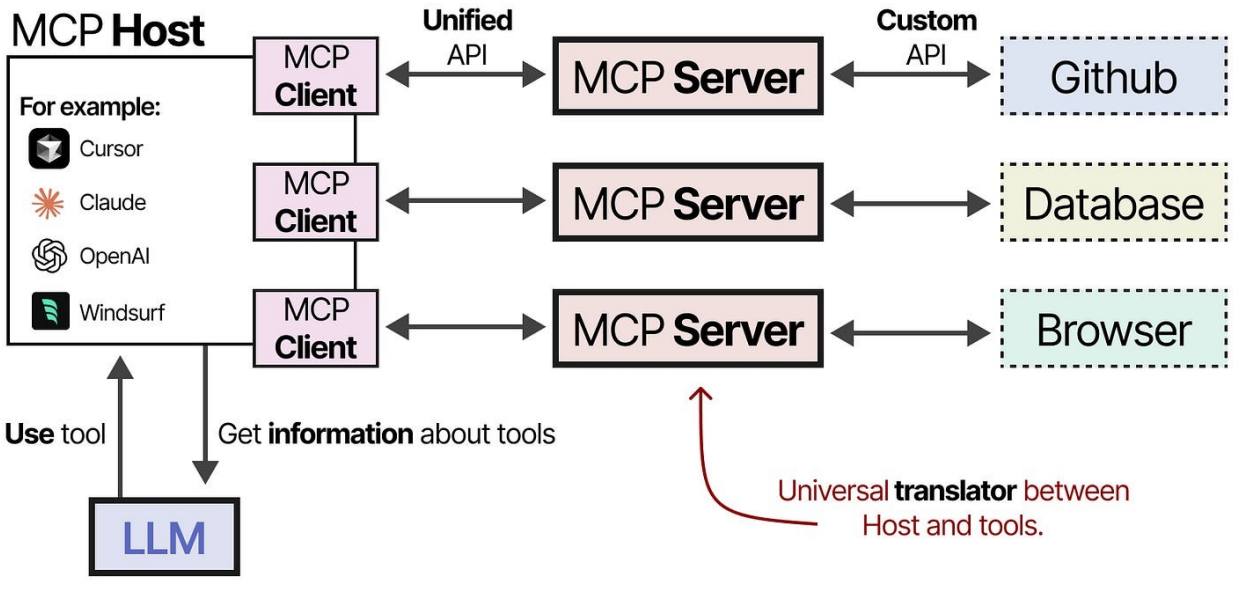
https://modelcontextprotocol.io/docs/getting-started/intro

MCP介绍

AI 应用（如 Claude Code 或 Claude Desktop）充当 **MCP 主机**，它通过为每个 MCP 服务器创建一个 **MCP 客户端**来建立连接，保持**客户端与服务器之间的一对一连接。**

角色	描述
MCP 主机 (Host)	协调和管理一个或多个 MCP 客户端的 AI 应用程序。
MCP 客户端 (Client)	维护与 MCP 服务器的连接，并从服务器获取上下文供主机使用。
MCP 服务器 (Server)	**提供上下文数据给 MCP 客户端的程序。**





MCP介绍

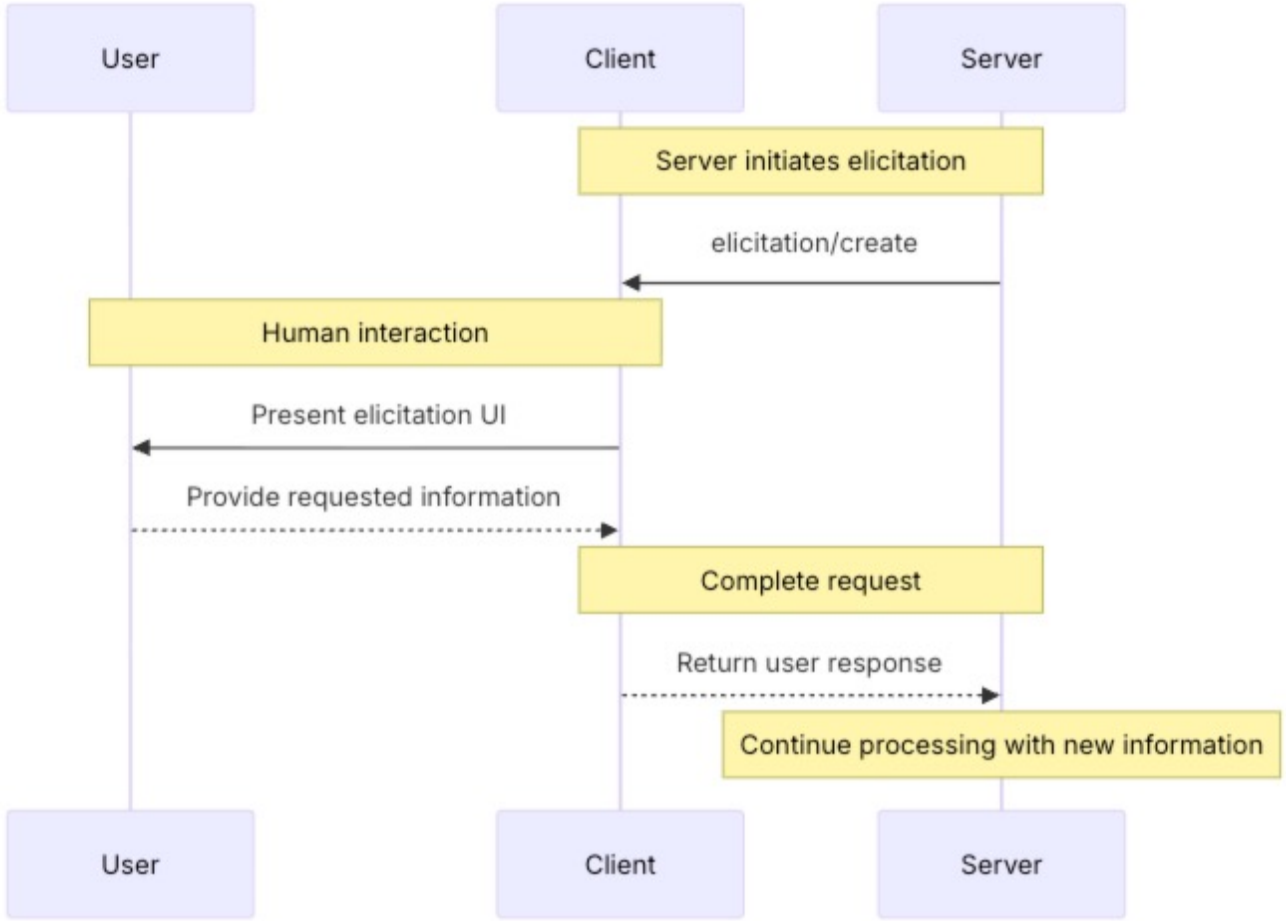
MCP 是一个**有状态协议**，需要进行生命周期管理。目的是在客户端和服务器之间**协商双方支持的功能(Capabilities)**，确保协议版本兼容，并交换身份信息。原语是 MCP 中最核心的概念，定义了客户端和服务器可以互相提供的内容。

原语	描述	示例方法
工具 (Tools)	AI 应用可调用以执行操作的可执行函数（例如文件操作、API 调用、数据库查询）。	tools/list (发现), tools/call (执行)
资源 (Resources)	提供上下文数据给 AI 应用的数据源（例如文件内容、数据库记录）。	resources/list (发现), resources/read (检索)
提示 (Prompts)	**帮助构建与语言模型交互的可重用模板（例如系统提示、少样本示例）。**	**prompts/list (发现), prompts/get (检索)**

MCP 支持两种传输机制：

- ✓ **Stdio 传输 (Stdio Transport):** 使用标准输入/输出流，用于同一机器上本地进程间的直接通信，具有最佳性能且无网络开销。
- ✓ **可流式 HTTP 传输 (Streamable HTTP Transport):** 使用 HTTP POST 进行客户端到服务器的消息传输，可选 Server-Sent Events 用于流式传输。支持远程服务器通信和标准 HTTP 认证方法（如 Bearer Tokens、OAuth）。

MCP介绍



- ✓ Host（主机）是最终用户直接交互的 **AI 应用程序。**
- ✓ Client（客户端）是Host 应用程序内部实例化和管理的**协议组件。**负责将 Server 的请求转发给 Host，并将 Host 的响应转发给 Server。
- ✓ Server（服务器)包含特定数据或功能的**外部程序**（可运行在本地或远程），提供 **工具**、**资源**、**提示** 给 Client 使用。

https://modelcontextprotocol.io/docs/learn/server-concepts
https://modelcontextprotocol.io/docs/learn/client-concepts

FastMCP使用

Run the Server

The simplest way to run your FastMCP server is to call its `run()` method. You can choose between different transports, like `stdio` for local servers, or `http` for remote access:

my_server.py (stdio) | my_server.py (HTTP)

```python
from fastmcp import FastMCP

mcp = FastMCP("My MCP Server")

@mcp.tool
def greet(name: str) -> str:
    return f"Hello, {name}!"

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

Call Your Server

Once your server is running with HTTP transport, you can connect to it with a FastMCP client or any LLM client that supports the MCP protocol:

my_client.py

```python
import asyncio
from fastmcp import Client

client = Client("http://localhost:8000/mcp")

async def call_tool(name: str):
    async with client:
        result = await client.call_tool("greet", {"name": name})
        print(result)

asyncio.run(call_tool("Ford"))
```

https://gofastmcp.com/getting-started/quickstart