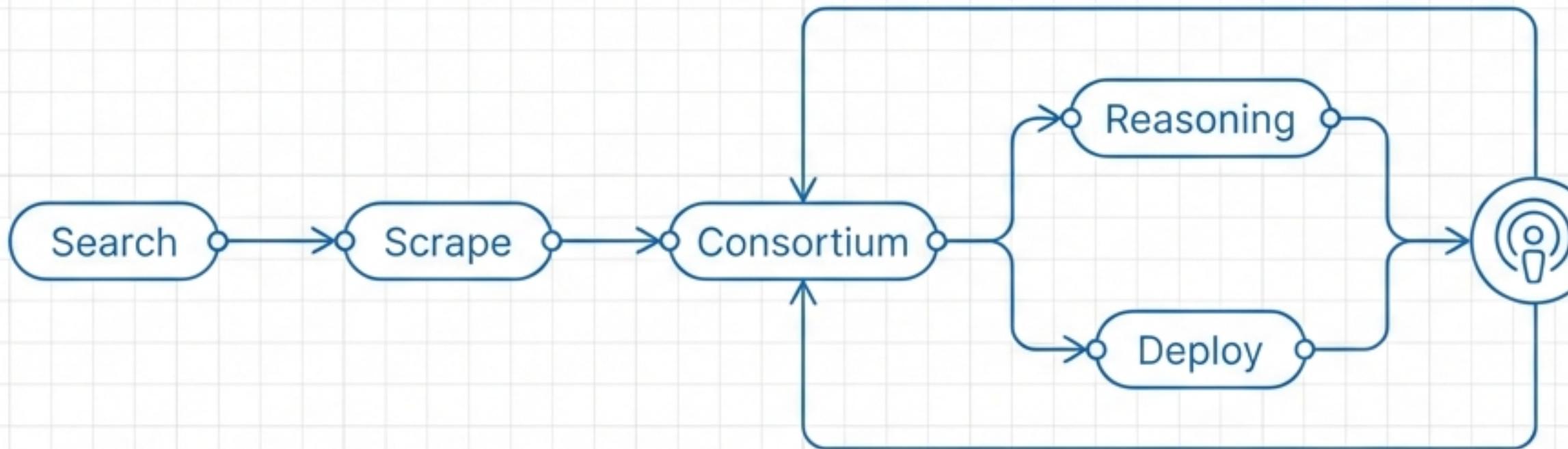


生产级 Agentic AI 工作流实战指南

针对设计、开发与部署的工程化最佳实践



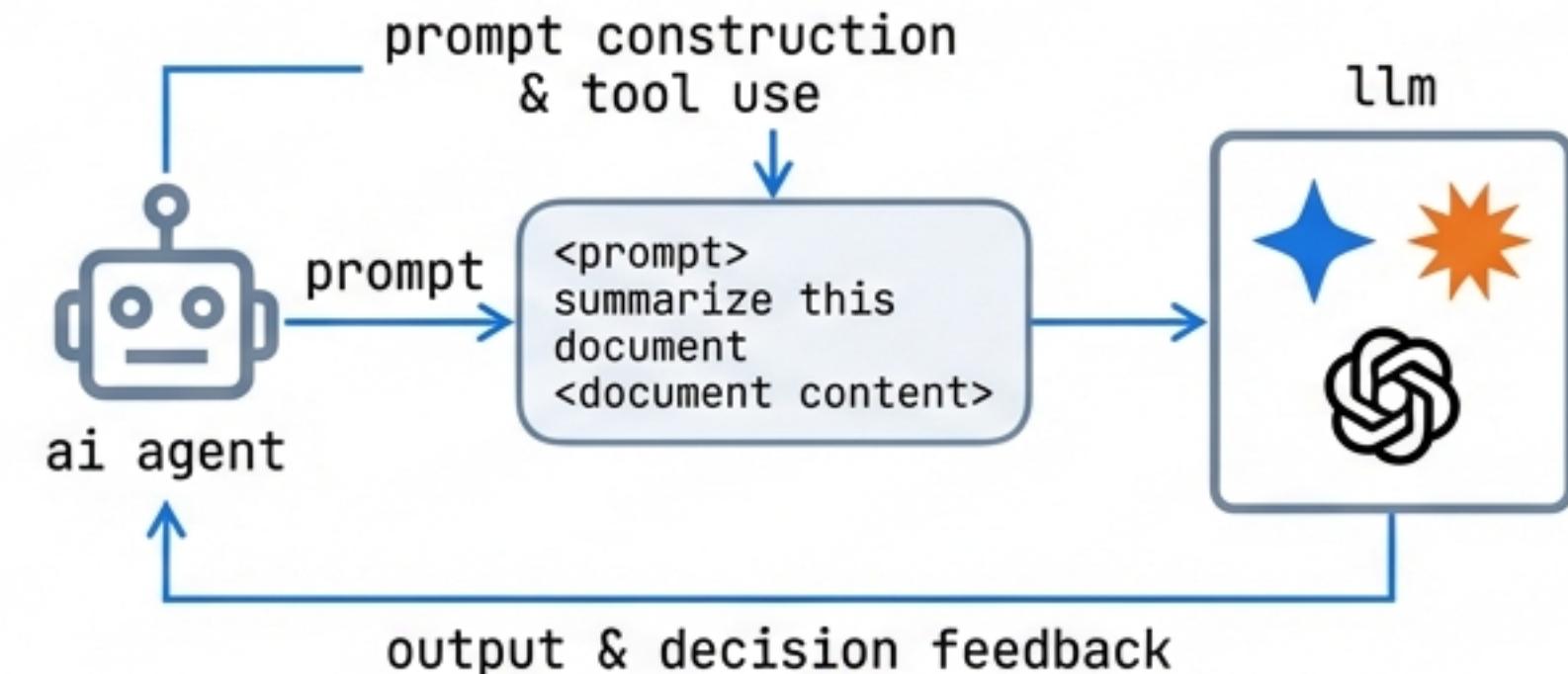
Based on research by Bandara et al. (2025): A Practical Guide for Designing, Developing, and Deploying Production-Grade Agentic AI Workflows

文档类型：工程手册
包含内容：9 项核心原则
案例研究：多模态播客生成

原型陷阱 (The Trap): 单体提示词



生产目标 (The Goal): 智能体工作流



- 简单易构建，但极易产生幻觉
- 缺乏错误恢复机制
- 无法处理多步骤复杂任务

- 自主决策与工具使用
- 具备自我修正能力的循环机制
- 支持多模态与外部系统集成

工程化四大挑战 (The 4 Challenges)

设计 (Design)

- 任务拆解与工具权衡

实现 (Implementation)

- Schema 管理与异构模型集成

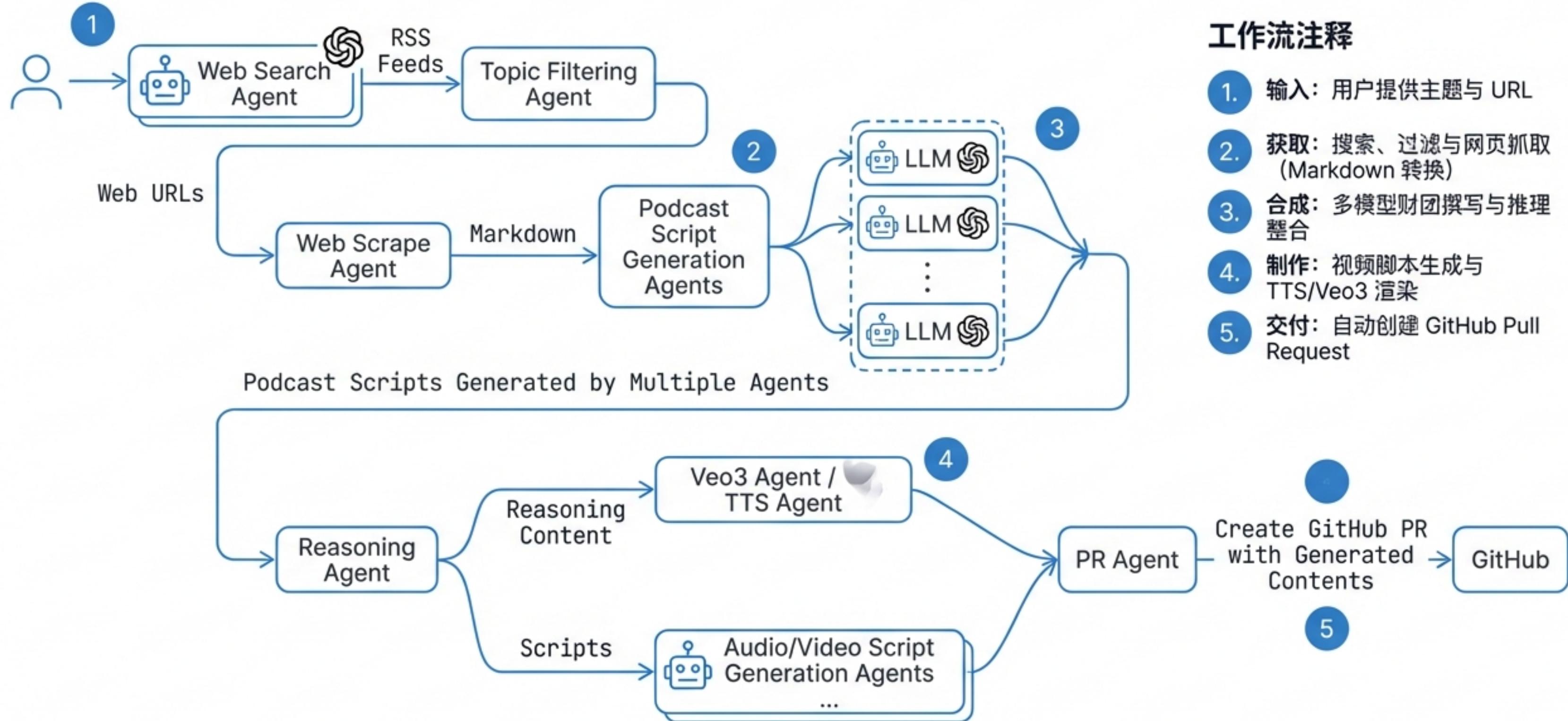
运维 (Operations)

- 并发控制与成本可观测性

部署 (Deployment)

- 版本控制与容器化编排

案例蓝图：多模态新闻播客生成工作流



设计原则 I：内部逻辑优先使用工具调用 (Tool Calls)

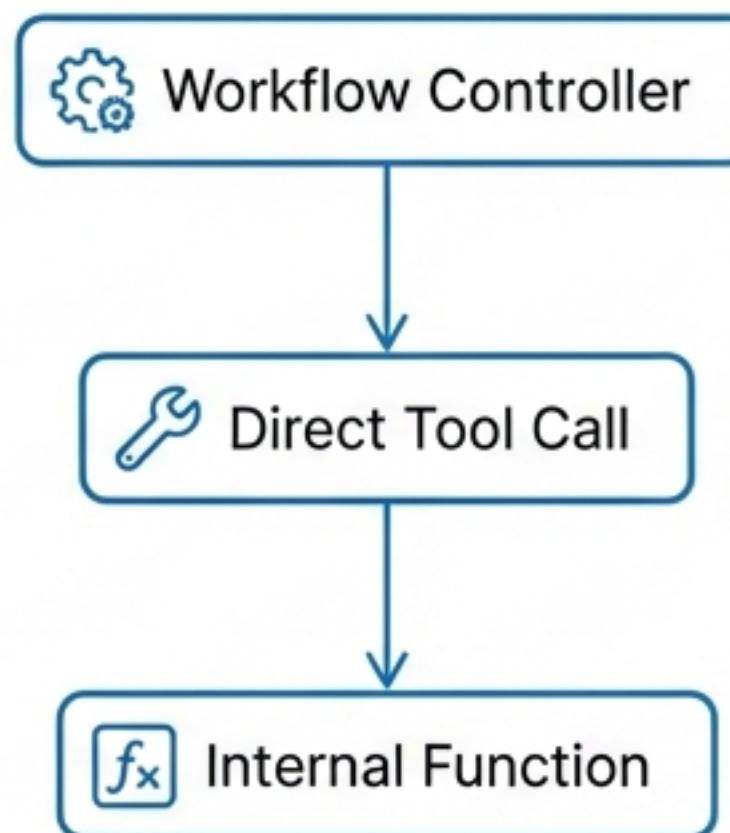
避免在工作流内部滥用 MCP 协议

错误范式：过度抽象的 MCP

```
1 github_pr_agent = Agent(  
2     name="Pull Request Creation Agent",  
3     instructions=  
4         "Create file with given file name and content(don't create file as base64 encoded) and push it  
5         does not exists create it from default branch first"  
6         "Then create a pull request, ensure the file is committed, pushed to the correct branch, and  
7         targeting the repository's default branch. branch."  
8     ),  
9  
10    async def main():  
11        # configure google mcp server and github mcp server  
12        async with HEPServerStdio(  
13            parameters={ ... }  
14            ) ...  
15        )  
16        google_search_agent.mcp_servers = [google_search_server]  
17        read_markdown_agent.mcp_servers = [google_search_server]  
18        github_pr_agent.mcp_servers = [github_server]  
19  
...  
...  
49        # observed error 1  
50    ! Note: File does not exist in branch, will create new file Error invoking MCP tool create_or_update  
51        hoa: Resource not found: Branch podcast-v1 not found  
52  
53        # observed error 2  
54    agents.exceptions.AagentsException: Error invoking MCP tool get_file_contents: Not Found: Resource  
      ref podcast-v1
```

使用 MCP 处理内部逻辑（如创建 PR）增加了不必要的抽象层，导致参数解析歧义和调试困难。

最佳实践：原生工具调用



保留 MCP 用于外部连接（如连接 IDE）。对于内部步骤，使用原生的 Tool Calls 以确保确定性和类型安全。

设计原则 II：确定的事情交给纯函数（Pure Functions）

并非所有步骤都需要 LLM 参与

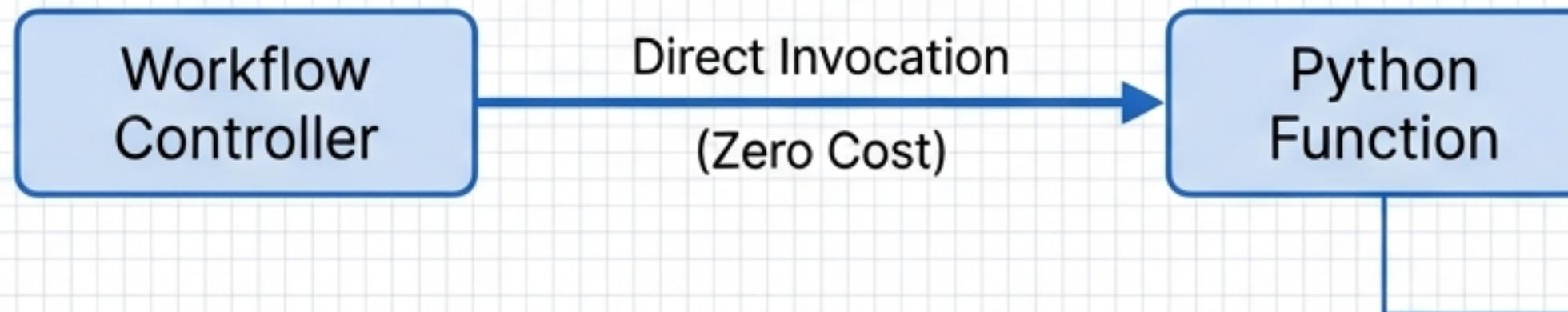
低效模式 (Agentic Tool Call)



- 浪费 Token 解析 JSON，且存在 Schema 错误风险。

```
# Function: GitHub PR creation
def create_github_pr(
    repo: str,
    branch: str,
    file_name: str,
    file_content: str,
    token: str = None,
    commit_message: str = "Automated PR from agentic workflow",
    base_branch: str = "master",
    pr_title: Optional[str] = None,
):
    """
    Create a file in a new or existing branch and open a Pull Request in GitHub. Requires a GitHub token with `repo` scope.
    """
    # implementation of pull request creation
    ...
```

高效模式 (Pure Function)



对于无需语义推理的任务（如 Git Commit, DB Write），直接调用代码函数。速度更快，100% 确定性。

原则：如果一个任务不需要‘思考’，就不要使用智能体。



设计原则 III：单一职责与原子化工具

One Agent, One Tool



混合了“规划”与“执行”，导致 JSON 格式错误和幻觉状态更新。



单一职责原则 (SRP)：将生成结构化数据的‘思考者’与执行 API 调用的‘执行者’分离。

设计原则 IV: KISS 原则与扁平化架构

过度工程化 (Over-Engineered)

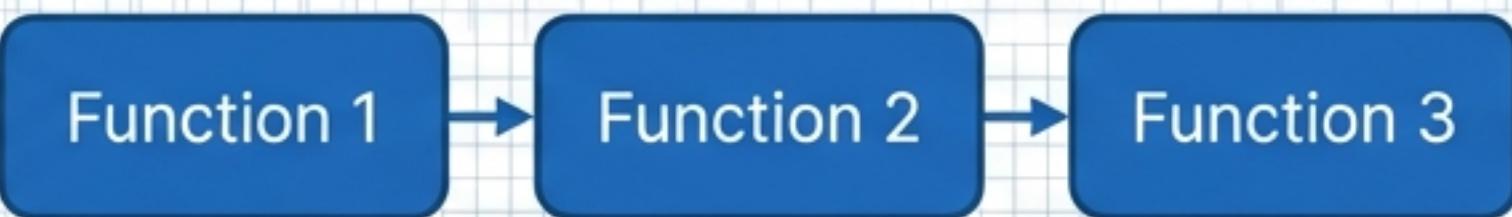


深层继承和复杂的微服务调用图使得系统难以调试，且 AI 辅助编程工具（如 Copilot）难以理解上下文。



“复杂性是可靠性的最大敌人。” 扁平且枯燥的代码更容易被人类维护，也更容易被 AI 编码助手理解。

生产级架构 (Production-Grade)



扁平的代码结构、简单的函数驱动设计。

开发原则 I：提示词的外部化管理



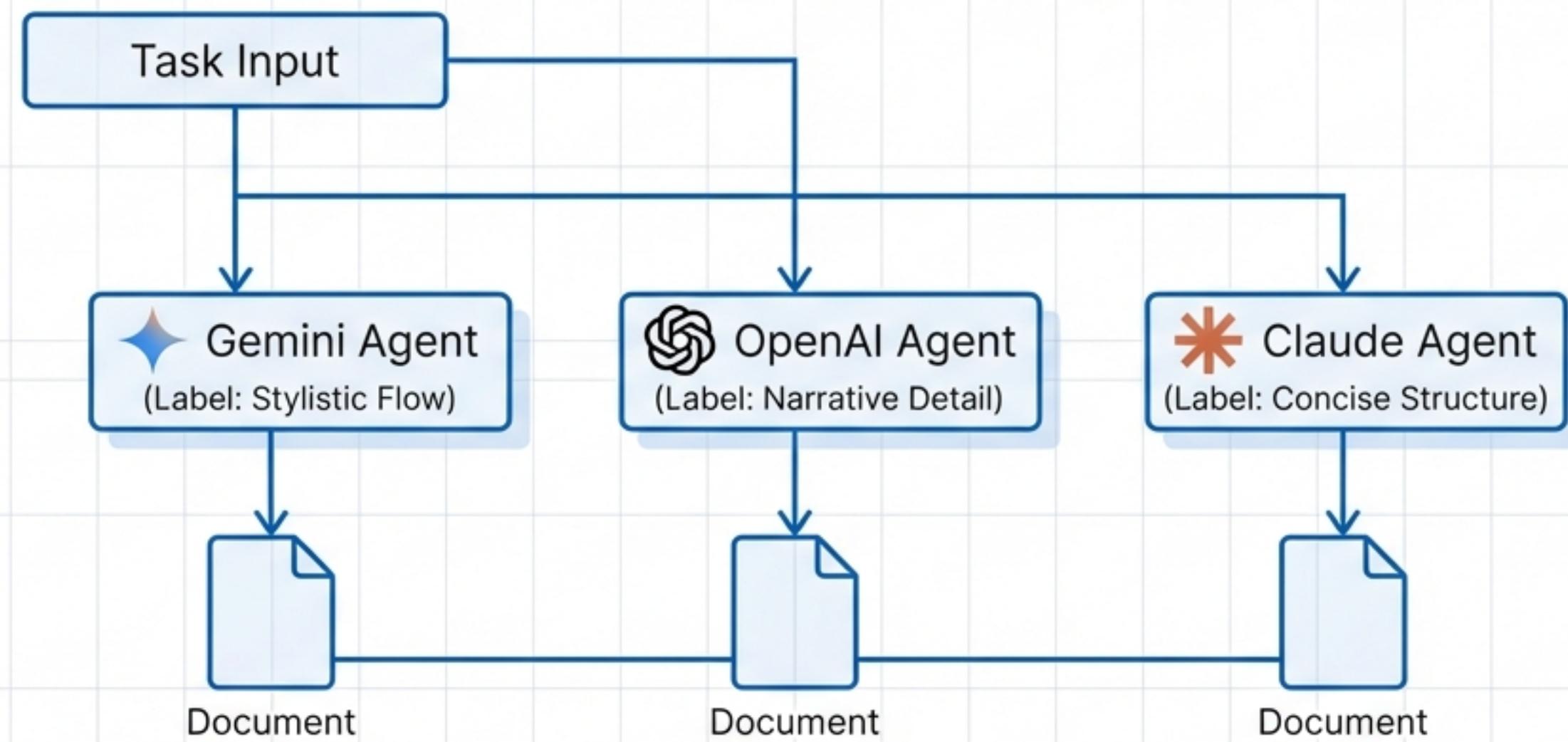
Prompts Repository
(.md / .txt)

```
instructions=load_prompt("podcast_generation_agent.txt")
```

- **非技术人员协作：**领域专家可直接修改 Markdown 提示词，无需触碰代码。
- **热更新：**无需重新编译即可调整 Agent 行为。
- **版本控制：**支持对提示词进行独立的 A/B 测试与回滚。

开发原则 II：多模型“财团”架构 (The Consortium)

利用模型多样性对抗单一偏见



策略：不依赖单一模型。通过异构模型组合（Ensemble），利用不同模型的训练偏差互补，提高系统的鲁棒性。

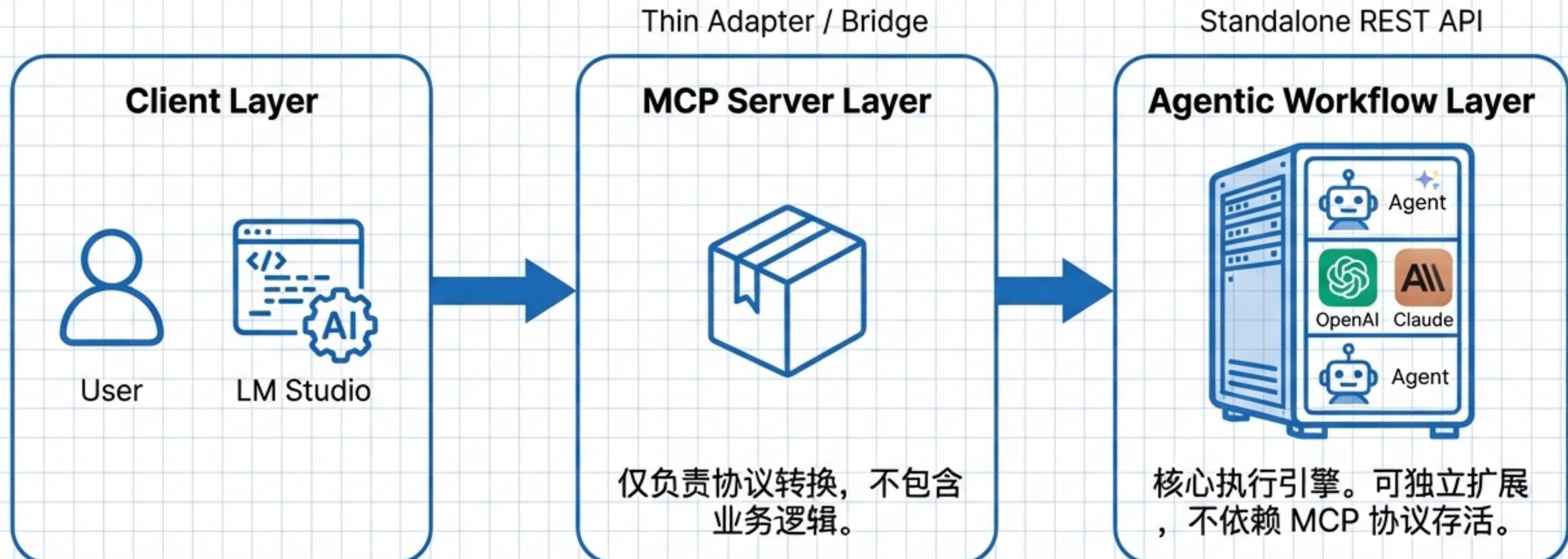
开发原则 III：推理智能体与自动审计

负责任 AI (Responsible AI) 的实现核心



推理层作为安全网，过滤掉仅在单一模型中出现的幻觉，确保最终输出的事实一致性。

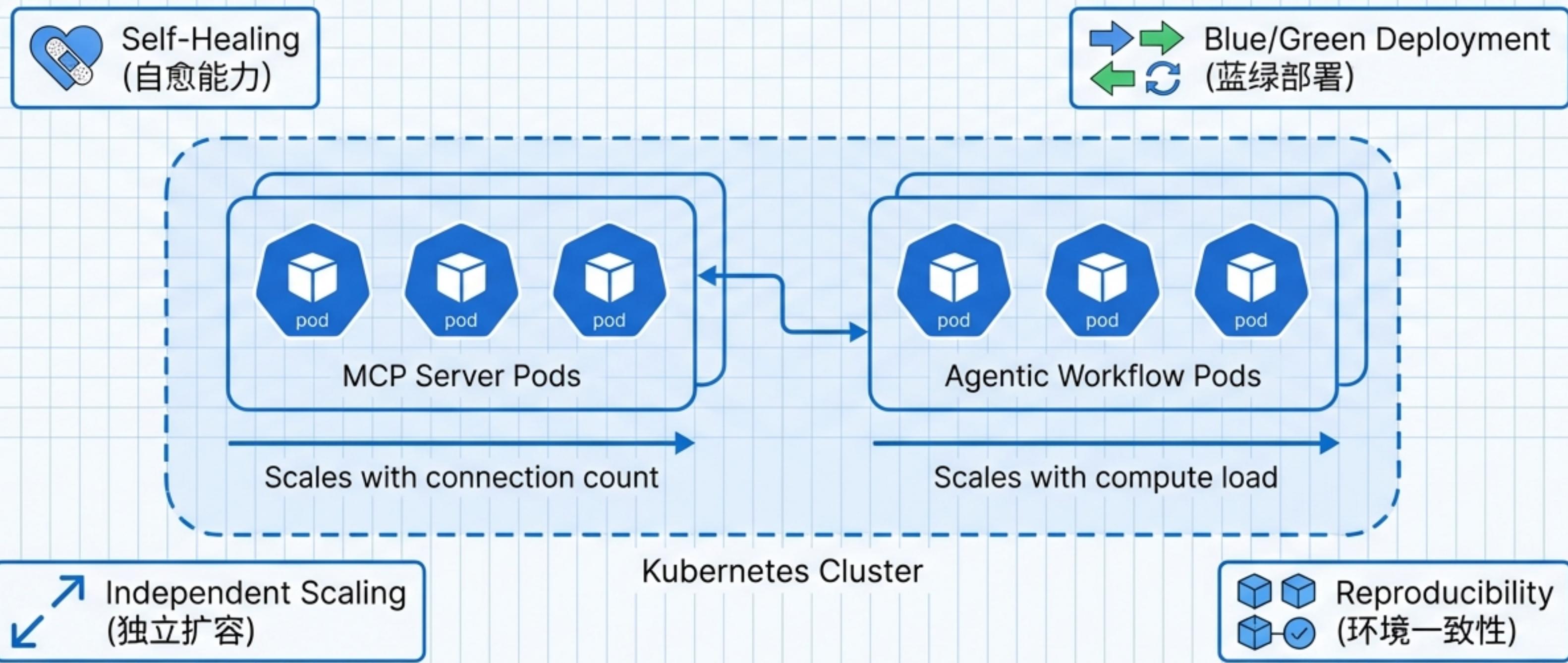
部署原则 I：工作流与 MCP 服务的解耦



解耦优势：工作流可独立于接口层进行迭代与扩容，MCP 仅作为一种可选的连接方式。

部署原则 II：容器化与 K8s 编排

从 Localhost 到 Production



成果验证：高质量多模态生成

Consolidated Script

...
15 - Alex: A familiar name is tied to this-reports link Commander Gregory Bovino to the effort; he previously led a controversial crackdown in the Chicago area.

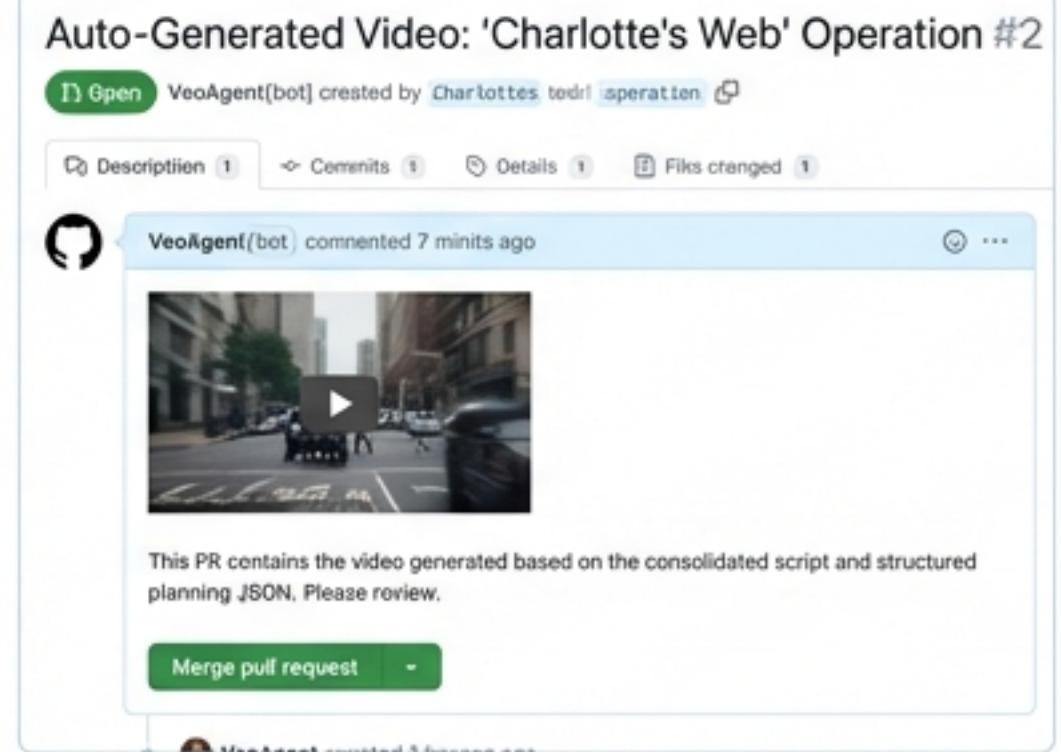
...
16 - Ben: Local Democratic officials, including Charlotte Mayor Vi Lyles, say the operation is causing unnecessary fear and uncertainty.

...

Structured Planning

```
{  
  "scene_description": "0-2s: Quick shot of a bustling city street, then a blurred, indistinct image of agents in uniform. Narration: DHS launched \"Charlotte's Web\" in Charlotte, targeting public safety threats...",  
  "camera_movement": "0-2s: Quick pan, then static blur. 2-4s: Dolly in, then static wide. 4-6s: Shaky handheld, quick cut...",  
}
```

Execution



Key Metric

Schema Validity: 100%
(Attributed to the separation of JSON Builder and Video Generator agents).

生产级 Agentic AI 工程化清单 (The Playbook)

Cheat Sheet

Tool-First Design

内部逻辑拒绝 MCP，使用原生工具。

Cheat Sheet

Pure Functions

确定性任务直接调用代码，不使用 LLM。

Cheat Sheet

One Agent, One Tool

降低认知负载，单一职责。

Cheat Sheet

Single Responsibility

分离规划 (Plan) 与执行 (Execute)。

Cheat Sheet

External Prompts

运行时加载提示词，解耦代码。

Cheat Sheet

Model Consortium

多模型协作对抗偏见。

Cheat Sheet

Reasoning Auditor

引入推理层进行最终事实核查。

Cheat Sheet

Decoupled Architecture

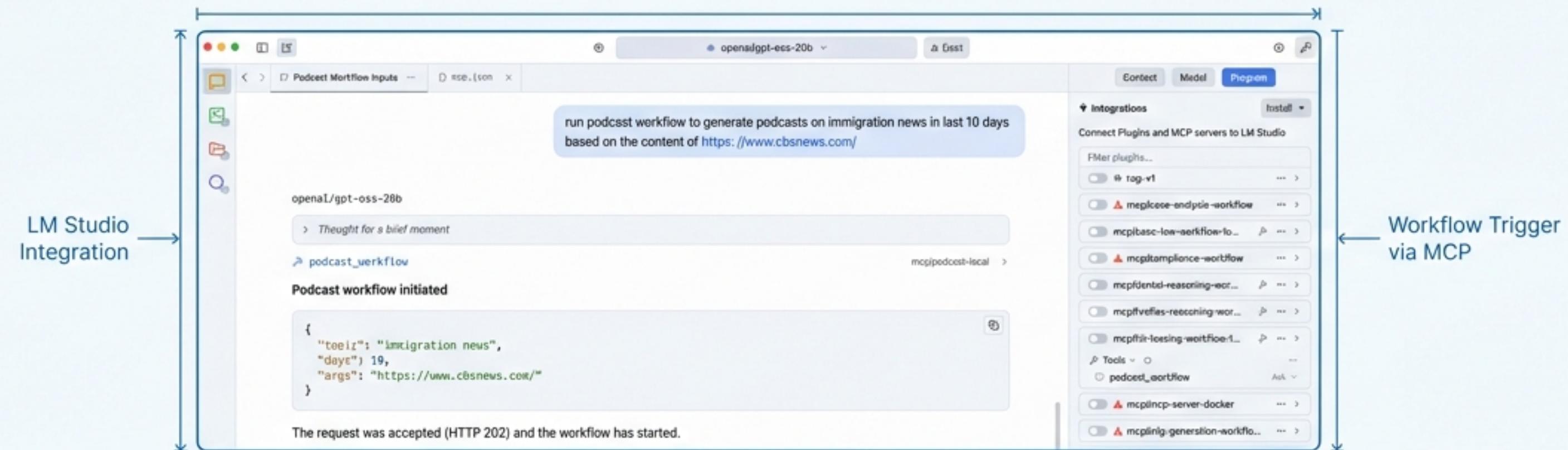
工作流 API 与 MCP 服务分离。

Cheat Sheet

Containerization

全链路 Docker 化与 K8s 编排。

下一步：无缝集成与生态互联



Agentic AI 的核心不在于提示词工程（Prompt Engineering），
而在于系统工程（Systems Engineering）。

Call to Action

Open Source Reference Implementation:

- Workflow: gitlab.com/rahasak-labs/podcast-workflow
- MCP Server: gitlab.com/rahasak-labs/podcast-workflow-mcp-server