解密一位活跃在Github上的非人类程序员

李明宇　中国科学院计算技术研究所

科技生态圈峰会 + 深度研习

——1000 + 技术团队的共同选择

KEYLINK ing

K+峰会

上海站
K+全球软件研发行业创新峰会
时间：2024.06.21-22

敦煌站
K+思考周®研习社
时间：2024.10.17-19

香港站
K+思考周®研习社
时间：2024.11.10-12

K+峰会详情

AiDD峰会

上海站
AI+研发数字峰会
时间：2024.05.17-18

北京站
AI+研发数字峰会
时间：2024.08.16-17

深圳站
AI+研发数字峰会
时间：2024.11.08-09

AiDD峰会详情

演讲嘉宾

李明宇

中科院计算所高通量计算机研究中心

处理器芯片国家重点实验室高级工程师

中国科学院计算技术研究所高级工程师，前深信服科技集团云计算首席专家。开源项目活跃开发者，20余年编程与软件开发经验，10余年云端系统研发与产业化工作经历。历任国家重点实验室研究组负责人，上市公司事业群技术总监、首席技术专家。李老师在理论研究和产业实践均有丰富的经验，为多家知名企业和单位落地云原生和数字化转型项目。荣获中国软件协会“优秀CTO”等多项荣誉，兼任中国新一代IT产业联盟分委会秘书长、全国高校人工智能与大数据创新联盟专家委员、北京开源创新委员会委员。

目录
CONTENTS

PART 01

有一位非人类程序员在 Github 上搞事情

Bug Fix

关于redis反弹shell #334

Closed **RuoJi6** opened this issue on Mar 5 · 2 comments · Fixed by #341

RuoJi6 commented on Mar 5

乌班图redis系统反弹shell，出现
CONFIG SET dir /var/spool/cron/
-ERR Changing directory: No such file or directory
Centos是/var/spool/cron/用户名
Ubuntu是/var/spool/cron/crontabs/用户名

DevBuddyConnor commented on Apr 13

针对Issue #334 关于redis反弹shell在Ubuntu系统中的路径问题，以下是一些建议的代码修改方式，以适应不同操作系统的路径差异问题。建议在 `writecron` 函数中增加操作系统路径的检测和相应的处理逻辑。

```
func writecron(conn net.Conn, host string) (flag bool, text string, err error) {
    flag = false
    // 尝试写入Ubuntu的路径
    _, err = conn.Write([]byte("CONFIG SET dir /var/spool/cron/crontabs/\r\n"))
    if err != nil {
        return flag, text, err
    }
    text, err = readreply(conn)
    if strings.Contains(text, "OK") {
        // 如果返回了"OK"，说明Ubuntu路径可用，接下来我们可以尝试写入cron
    } else {
        // 如果没有返回"OK"，可能是CentOS，尝试CentOS的路径
        _, err = conn.Write([]byte("CONFIG SET dir /var/spool/cron/\r\n"))
        if err != nil {
            return flag, text, err
        }
        text, err = readreply(conn)
```

历史对话

针对提到的redis反弹shell问题，最相关的代码文件应该是：

- `Plugins/redis.go`

这个文件可能包含了操作redis的相关代码逻辑，包含反弹shell的实现。对于乌班图(Ubuntu)和CentOS系统在路径上的不同，解决方案可能需要在代码中加入系统检测逻辑，以确定正确的路径。

建议的修改方案

我们可以按照以下步骤修改 `redis.go` 文件：

1. 检测操作系统类型。
2. 根据操作系统类型设置正确的路径。
3. 适配redis命令以使用正确的路径。

我将获取 `redis.go` 文件的内容，以便提供具体的修改建议。

根据 `redis.go` 的文件内容，特别是在处理反弹shell时，可以看到 `writecron` 函数是用于写入cron任务的函数，其中有这样一行代码负责设置cron的目录：

```
_, err = conn.Write([]byte("CONFIG SET dir /var/spool/cron/\\r\\n"))
```

为了解决Ubuntu系统路径问题，我们可以增加一个步骤来检查操作系统，并相应地设置正确的路径。但由于Fscan是一个远程扫描工具，我们不能直接使用Go的标准库来检查目标系统的类型。解决方案可以是尝试写入Ubuntu的cron路径，如果失败，再尝试CentOS的路径。

下面是一个修改后的 `writecron` 函数的示例代码：

```
func writecron(conn net.Conn, host string) (flag bool, text string, err error) {
    flag = false
    // 尝试写入Ubuntu的路径
    _, err = conn.Write([]byte("CONFIG SET dir /var/spool/cron/crontabs/\\r\\n"))
    if err != nil {
        return flag, text, err
```

PR Review

Review Comment for PR #37 and PR #39 in Context of Issue #36

Summary of Changes in PR #37 and PR #39:

- PR Fix #36 and openai lib incompatibility issue. #37 updates Flask to 3.0.0 and modifies the `requirements.txt` file for resolving compatibility issues with Flask and Werkzeug, as well as updating other dependencies.
- PR Add healthcheck to the application code and k8s yamls #39 introduces a health check endpoint (`/healthz`) and updates the Kubernetes YAML files to include readiness probes for this endpoint.

Evaluation:

- **PR** Fix #36 and openai lib incompatibility issue. #37: Successfully addresses the compatibility issues stated in Issue 代码更新后，线上pr-review-gpt不能正常工作 #36. The update to Flask 3.0.0 is a crucial step in resolving the Werkzeug dependency conflict.
- **PR** Add healthcheck to the application code and k8s yamls #39: The introduction of a health check endpoint is a significant improvement, particularly in the context of Kubernetes deployments. It aids in managing rolling updates and service availability.

Suggestions for Further Improvements:

1. **Enhance Health Check (PR** Add healthcheck to the application code and k8s yamls #39**)**: Expand the health check functionality to include database connectivity checks and OpenAI API availability. This can be done by adding code to the `/healthz` endpoint to test MongoDB connection and a basic OpenAI API functionality check. This will ensure the service's key components are operational before declaring the service "healthy".

```python
@app.route('/healthz')
def healthz():
    # Check MongoDB Connection
    try:
        client.server_info()  # Attempts to get server info to check connection
    except Exception as e:
        return f"MongoDB Connection Failed: {e}", 500

    # Check OpenAI API Key availability and basic functionality
    try:
        openai.api_key = os.getenv("OPENAI_API_KEY")
        openai.Completion.create(model="text-davinci-003", prompt="Hello", max_tokens=5)
    except Exception as e:
        return f"OpenAI API Check Failed: {e}", 500

    return "Healthy", 200
```

2. **Adjust Kubernetes Probes (PR** Add healthcheck to the application code and k8s yamls #39**)**: Given the more comprehensive health checks, review and possibly adjust the readiness probe parameters in the Kubernetes configuration. Consider increasing `timeoutSeconds` to accommodate the additional time needed for the extended checks, adjusting `initialDelaySeconds` to ensure the service has enough time to establish necessary connections, and revising `periodSeconds` and `failureThreshold` to balance responsiveness with stability.

```yaml
readinessProbe:
  httpGet:
    path: /healthz
    port: 5000
    scheme: HTTP
  initialDelaySeconds: 10  # Increased delay
  periodSeconds: 5         # Less frequent checks
  successThreshold: 1
  failureThreshold: 3      # Higher tolerance for intermittent failures
  timeoutSeconds: 5        # Increased timeout
```

Conclusion:
The combination of PR #37 and PR #39, along with the proposed enhancements to the health check, forms a comprehensive solution to Issue #36. The updates effectively address both the compatibility issue and the operational stability concerns in a Kubernetes environment. I recommend merging these PRs with the suggested enhancements to the health check for a more robust and reliable deployment.

Review Comment for PR #37 and PR #39 in Context of Issue #36

Summary of Changes in PR #37 and PR #39:

- PR Fix #36 and openai lib incompatibility issue. #37 updates Flask to 3.0.0 and modifies the `requirements.txt` file for resolving compatibility issues with Flask and Werkzeug, as well as updating other dependencies.
- PR Add healthcheck to the application code and k8s yamls #39 introduces a health check endpoint (`/healthz`) and updates the Kubernetes YAML files to include readiness probes for this endpoint.

Evaluation:

- **PR** Fix #36 and openai lib incompatibility issue. #37: Successfully addresses the compatibility issues stated in Issue 代码更新后，线上pr-review-gpt不能正常工作 #36. The update to Flask 3.0.0 is a crucial step in resolving the Werkzeug dependency conflict.
- **PR** Add healthcheck to the application code and k8s yamls #39: The introduction of a health check endpoint is a significant improvement, particularly in the context of Kubernetes deployments. It aids in managing rolling updates and service availability.

Suggestions for Further Improvements:

1. **Enhance Health Check (PR** Add healthcheck to the application code and k8s yamls #39**)**: Expand the health check functionality to include database connectivity checks and OpenAI API availability. This can be done by adding code to the `/healthz` endpoint to test MongoDB connection and a basic OpenAI API functionality check. This will ensure the service's key components are operational before declaring the service "healthy".

```python
@app.route('/healthz')
def healthz():
    # Check MongoDB Connection
    try:
        client.server_info()  # Attempts to get server info to check connection
    except Exception as e:
        return f"MongoDB Connection Failed: {e}", 500

    # Check OpenAI API Key availability and basic functionality
    try:
        openai.api_key = os.getenv("OPENAI_API_KEY")
        openai.Completion.create(model="text-davinci-003", prompt="Hello", max_tokens=5)
    except Exception as e:
        return f"OpenAI API Check Failed: {e}", 500

    return "Healthy", 200
```

2. **Adjust Kubernetes Probes (PR** Add healthcheck to the application code and k8s yamls #39**)**: Given the more comprehensive health checks, review and possibly adjust the readiness probe parameters in the Kubernetes configuration. Consider increasing `timeoutSeconds` to accommodate the additional time needed for the extended checks, adjusting `initialDelaySeconds` to ensure the service has enough time to establish necessary connections, and revising `periodSeconds` and `failureThreshold` to balance responsiveness with stability.

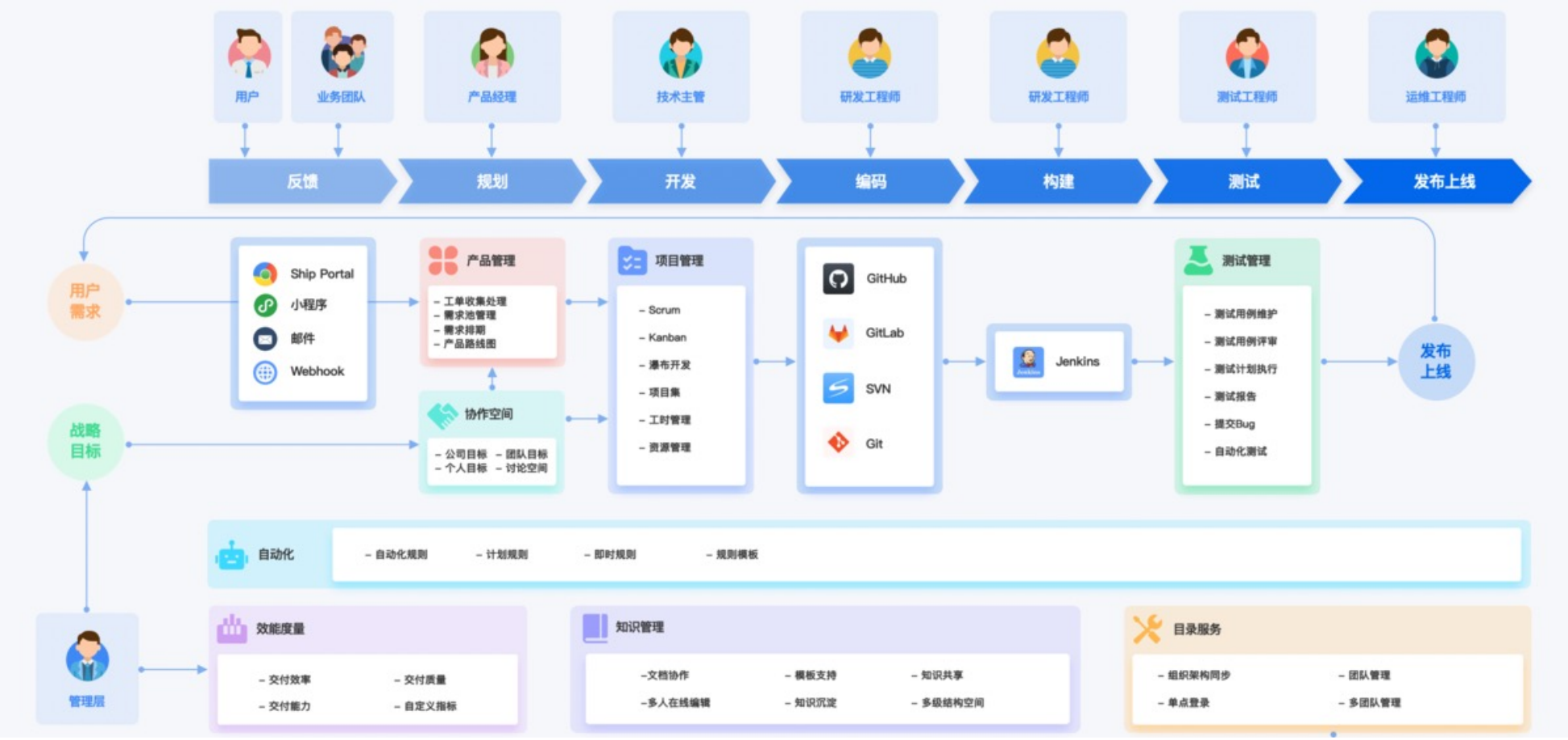
```yaml
readinessProbe:
  httpGet:
    path: /healthz
    port: 5000
    scheme: HTTP
  initialDelaySeconds: 10  # Increased delay
  periodSeconds: 5         # Less frequent checks
  successThreshold: 1
  failureThreshold: 3      # Higher tolerance for intermittent failures
  timeoutSeconds: 5        # Increased timeout
```

Conclusion:
The combination of PR #37 and PR #39, along with the proposed enhancements to the health check, forms a comprehensive solution to Issue #36. The updates effectively address both the compatibility issue and the operational stability concerns in a Kubernetes environment. I recommend merging these PRs with the suggested enhancements to the health check for a more robust and reliable deployment.

PART 02
TA能帮我们做什么？

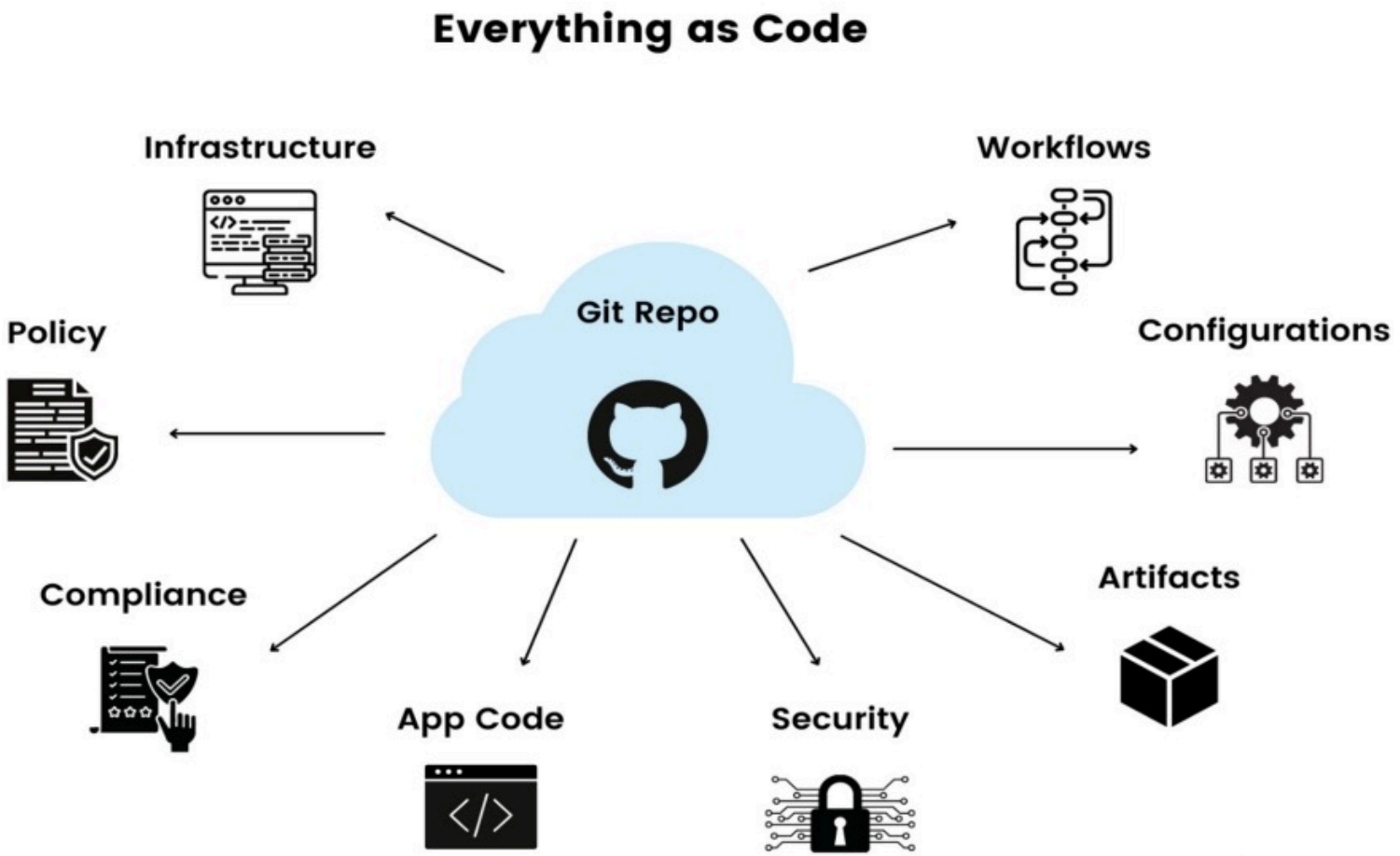
50%的代码自动生成能提高50%的生产力吗?

需求分析、设计规划、编程实现、测试验证、代码审查、部署上线与运维、监测与反馈



基于大语言模型的智能体深度参与软件研发过程

需求分析、设计规划、编程实现、测试验证、代码审查、部署上线与运维、监测与反馈



Code everything

LLM everything

典型场景与痛点问题

1. **项目代码的自主掌控的挑战：**基于开源项目实现自主可控以及委托乙方开发的代码，如何真正读懂和实际掌握，实现资产化，并有能力进一步修改，是行业普遍痛点；
2. **AI代码补全工具与解决实际软件开发问题的差距：**目前的AI工具助力程序员在开发环境中做代码补全，但实际软件项目中要实现一个功能或修复一个bug，花在分析问题、确定修改位置和方案的时间占绝大部分；
3. **代码评审工作对核心程序员的消耗与长期软件可维护性的权衡：**代码评审在工作压力大的情况下被忽视，坚持代码评审可能阻塞工作进展，损害短期利益，但放松代码评审可能导致日后软件的可维护性下降，损害长期利益。
4. …… 其他，例如问题反馈的管理：反馈上来的诸多Issue可能是bug可能是其他，可能重复或者仅仅是有用户提问）

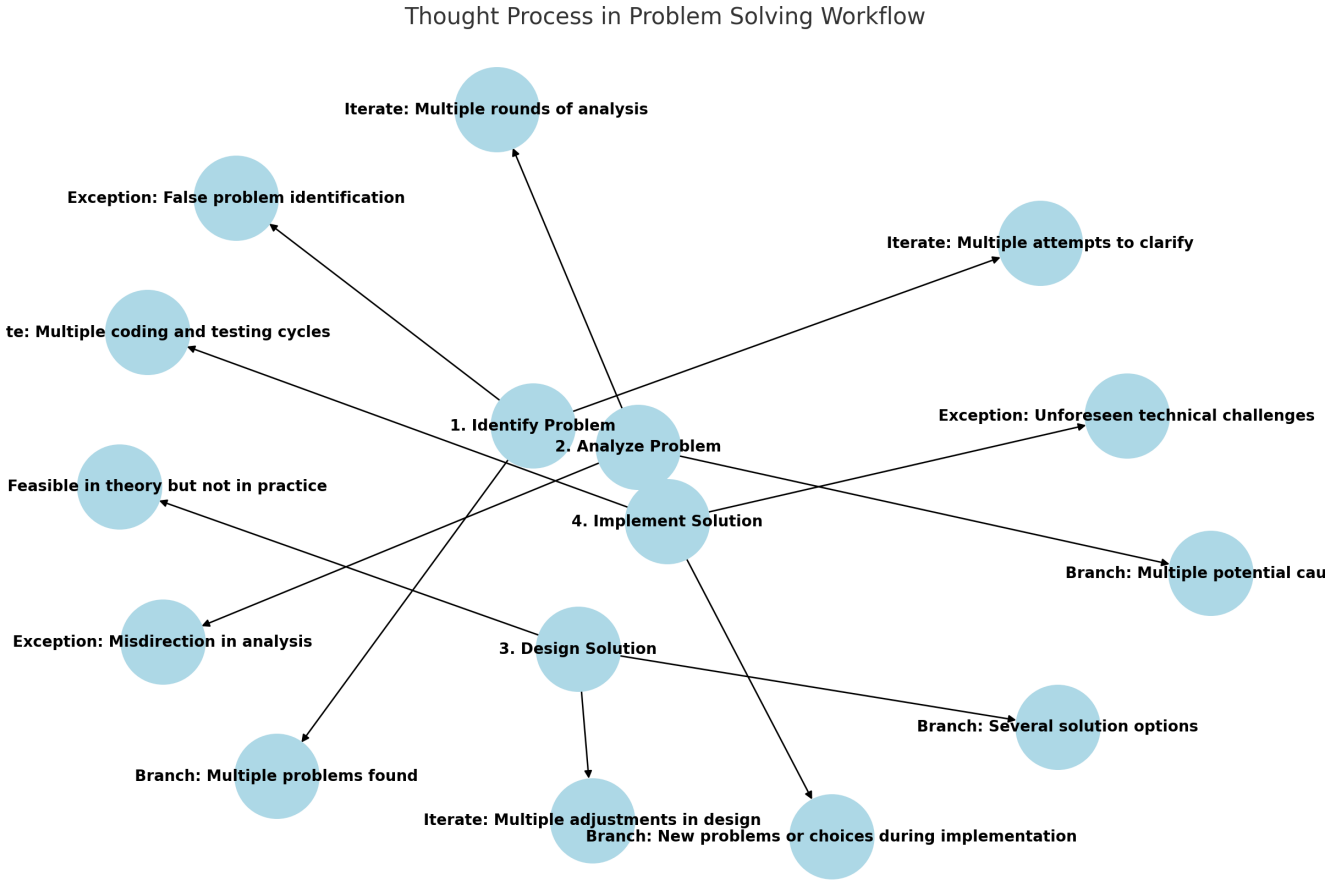
效果如何？不看广告看疗效

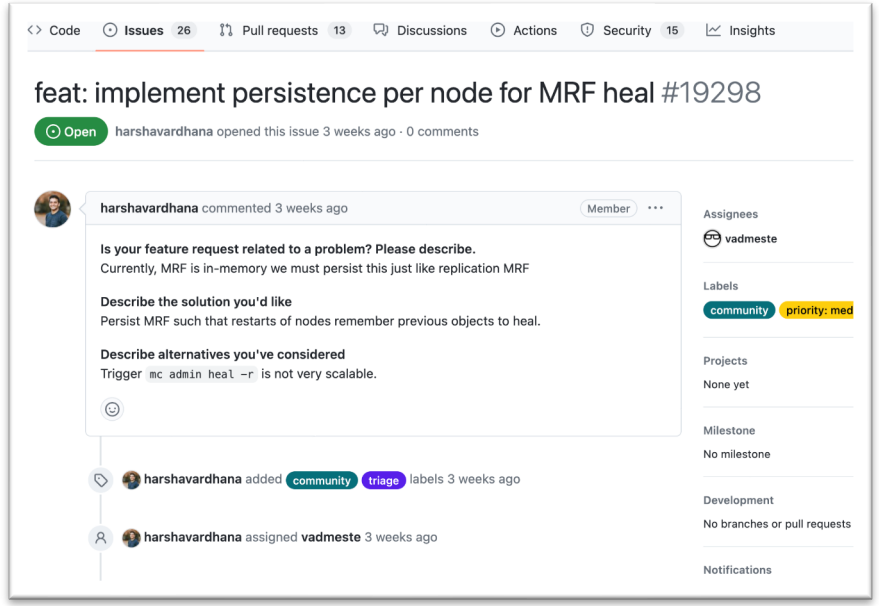
Live Demo

PART 03
实现思路与原理解密

实现思路与原理解密：以 issue fix 为例

- 程序员解决一个issue，一般的工作流是怎样的?



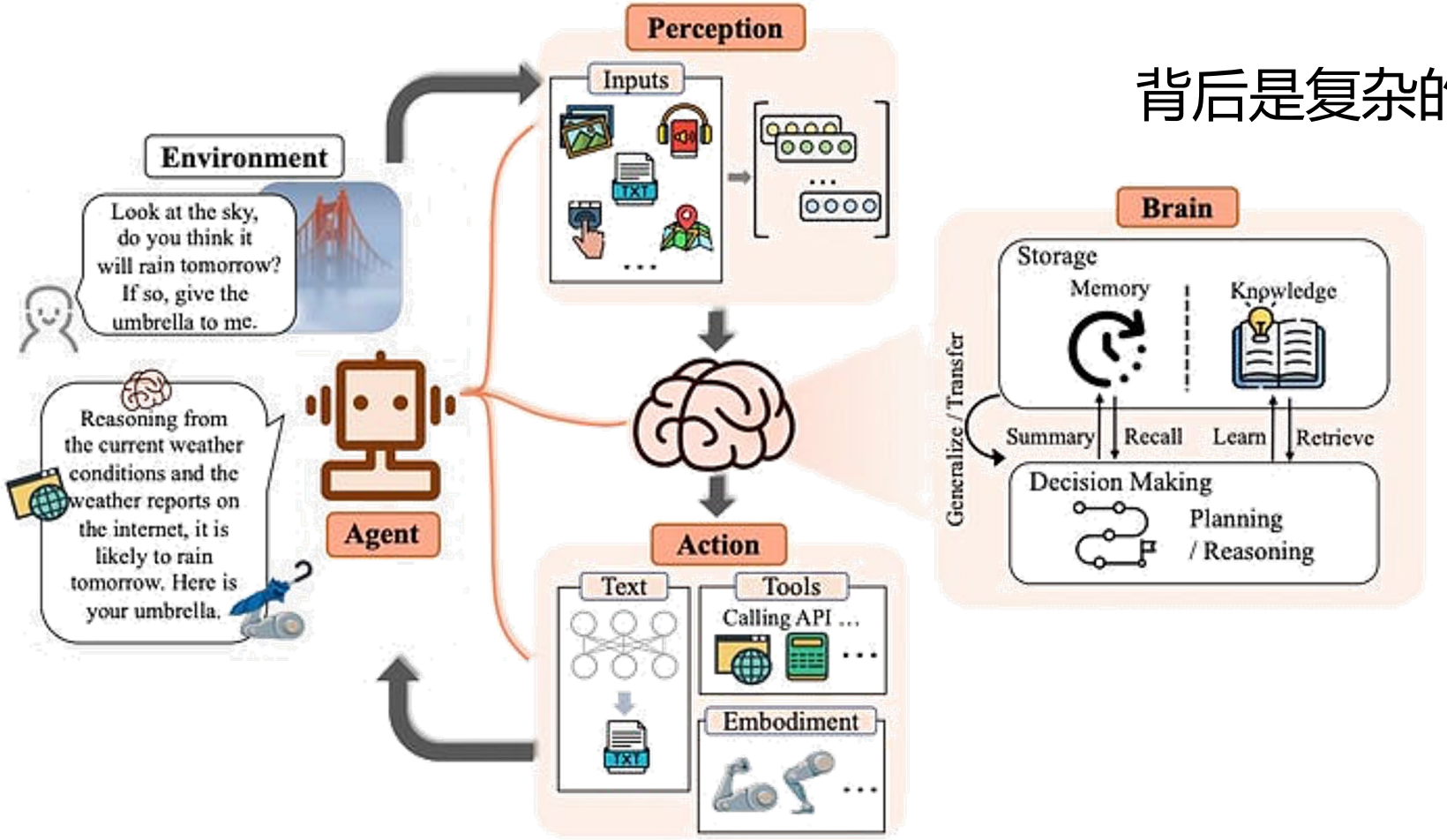


1. 理解识别问题
2. 查看相关代码和文档，分析原因
3. 提出解决思路/设计解决方案
4. 撰写/修改代码，实现解决方案

实现思路与原理解密：以 issue fix 为例

- AI Agent（智能体）

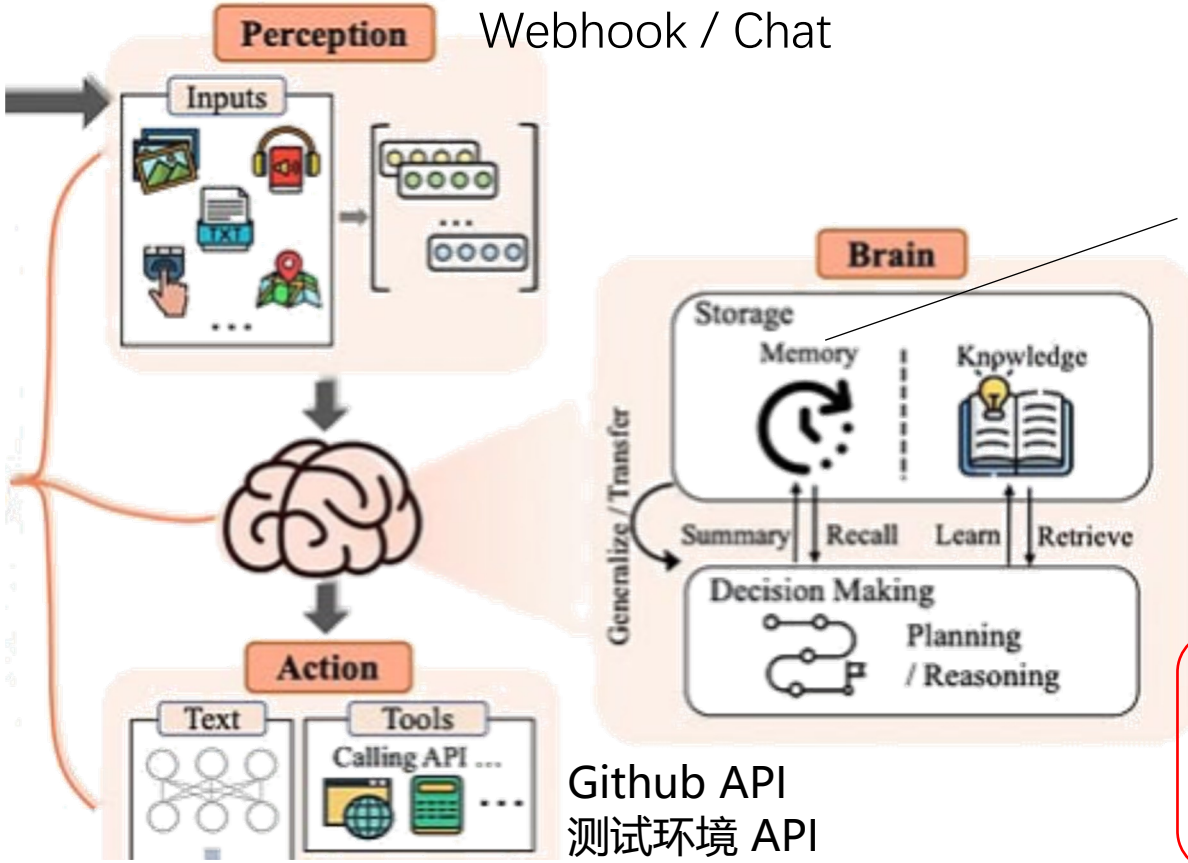
看似简单的
一问一答



背后是复杂的机制

实现思路与原理解密：以 issue fix 为例

Issue Fixing智能体



Webhook / Chat

关于该Issue的
对话/分析历史记录

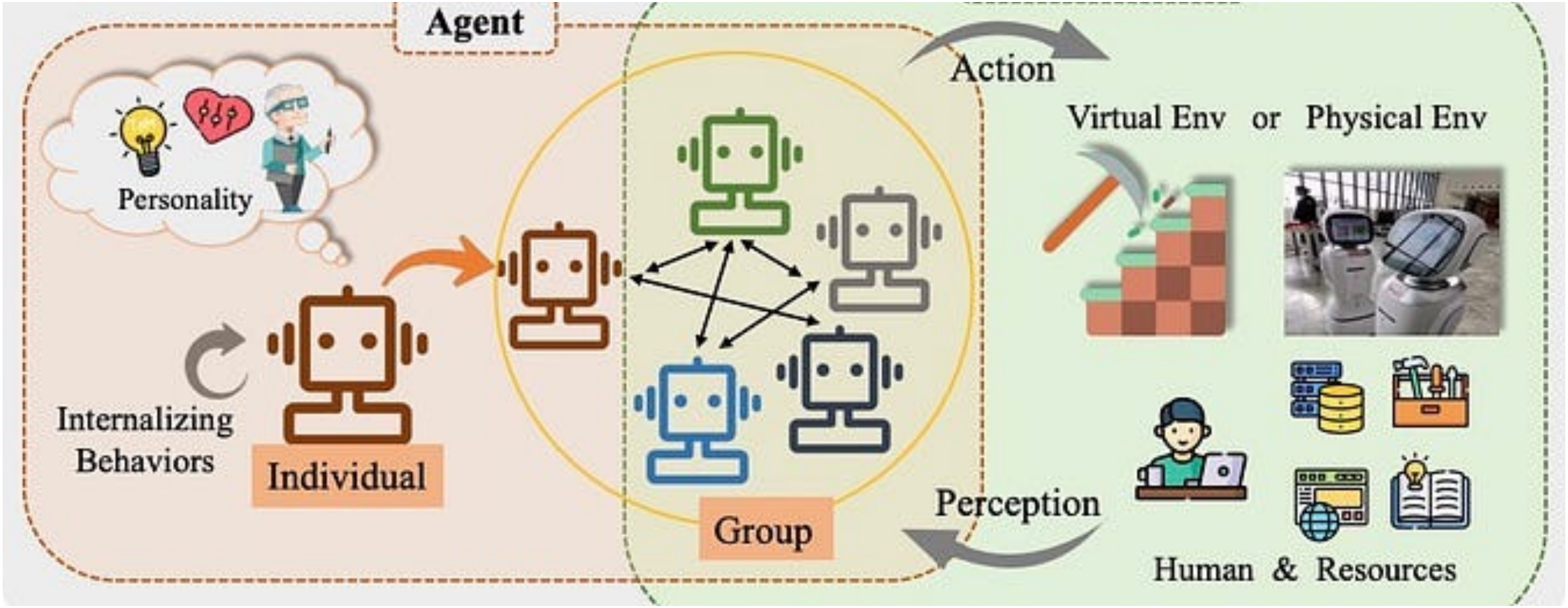
代码库 / 文档库 / 知识库
（RAG）

Github API
测试环境 API

前置：API调用与结果解析
主线：先输出解决方案，再生成代码
分支1：是否缺少信息需要用户澄清
分支2：是否需要用户确认方案后再生成代码

克服Agent死循环/跑飞问题

实现思路与原理解密：以 issue fix 为例



PART 04
更重要的是 —— 人

更重要的是 —— 人

不同的程序员应用AI Agents工具效果差别很大

对程序员的招聘和考核标准可能的变化

DevAgents 不取代程序员，也不贩卖焦虑，帮助程序员更好地思考和表达

帮助企业软件研发在AI时代持续成功

培训

编程Copilot实践 + DevAgents助理软件研发提效

咨询

持续咨询顾问，切实结合实际业务提升研发效率

工具

DevAgents 企业版可定制对接 Jira、Confluence、Gitlab

……

科技生态圈峰会 + 深度研习

——1000 + 技术团队的共同选择

KEYLINK ing

K+峰会

上海站
K+全球软件研发行业创新峰会
时间：2024.06.21-22

敦煌站
K+思考周®研习社
时间：2024.10.17-19

香港站
K+思考周®研习社
时间：2024.11.10-12

K+峰会详情

AiDD峰会

上海站
AI+研发数字峰会
时间：2024.05.17-18

北京站
AI+研发数字峰会
时间：2024.08.16-17

深圳站
AI+研发数字峰会
时间：2024.11.08-09

AiDD峰会详情

AiDD

THANKS