

AI 驱动 软件研发 全面进入数字化时代

中国·深圳 11.24-25

AI+
software
Development
Digital
summit



基于GPT的代码PR review助手开发与应用

李明宇 中国科学院计算技术研究所

▶ 演讲嘉宾



李明宇

中科院计算所高级工程师 / “编程老师傅”

- 前深信服科技集团云计算首席专家
- 中科院计算所高级工程师，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

1. 背景与痛点 —— 让人又爱又恨的 Code Review 环节
2. Code-Chat-Reviewer 开发历程与实现原理
3. 应用效果与优缺点
4. 11月6日以来的新进展
5. 总结与展望

目前的PR Review生成效果

Merged Add healthcheck to the application code and k8s yamls #39
Li-Mingyu merged 2 commits into main from health-check yesterday

AssistantRHINO commented 3 minutes ago

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

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

- PR #37 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 #39 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 #37 Fix #36 and openai lib incompatibility issue. #37: Successfully addresses the compatibility issues stated in Issue #36. The update to Flask 3.0.0 is a crucial step in resolving the Werkzeug dependency conflict.
- PR #39 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:

- Enhance Health Check (PR #39 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".

```
@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
```

- Adjust Kubernetes Probes (PR #39 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.

```
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.

▶ 目前的PR Review生成效果

Merged

Add healthcheck to the application code and k8s yamls #39
LI-Mingyu merged 2 commits into `main` from `health-check` yesterday



AssistantRHINO commented 3 minutes ago

Member ...

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

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".

```
@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.



```
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.

```
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.

目前的PR Review生成效果

对比：过去（4月份）

【AI审查】

【更改】

- 添加了一个新的类`CustomDiffViolations`，扩展了`DiffViolations`并覆盖了其`_init`
- `CustomDiffViolations`类还添加了一个新方法`_custom_diff_violations`。
- 这些更改与计算结构化数据类型（如列表或字典）中的更改的代码相关。



【建议】

- 通过遵循现有代码库的相同命名约定，确保一致性和可读性。
- 使用更具描述性的变量名称来提高代码清晰度。
- 彻底测试这些更改，以确保它们不会引入任何意外的副作用或破坏现有功能。
- 将中文注释翻译为英文，以便更好地访问。

【结论】

总的来说，这些更改似乎是合理且有良好文档记录的。但是，重要的是要确保这些更改不会引入任何意外的副作用或破坏现有功能。合并到主代码库之前，应进行彻底的测试和审查过程。

【行动】

在将这些更改合并到主代码库之前，彻底测试并解决提供的建议。

【其他】

为了使审查更加彻底，提供更改的背景和目的很重要。此外，以团队可以理解的语言提供注释对于清晰度和可访问性很重要。

Merged Add healthcheck to the application code and k8s yamls #39
LI-Mingyu merged 2 commits into main from health-check yesterday

AssistantRHINO commented 3 minutes ago

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

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

- PR #37: 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 #39: 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 #37: Fix #36 and openai lib incompatibility issue. #37: Successfully addresses the compatibility issues stated in Issue #36. The update to Flask 3.0.0 is a crucial step in resolving the Werkzeug dependency conflict.
- PR #39: 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:

- Enhance Health Check (PR #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".

```
@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
```

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.

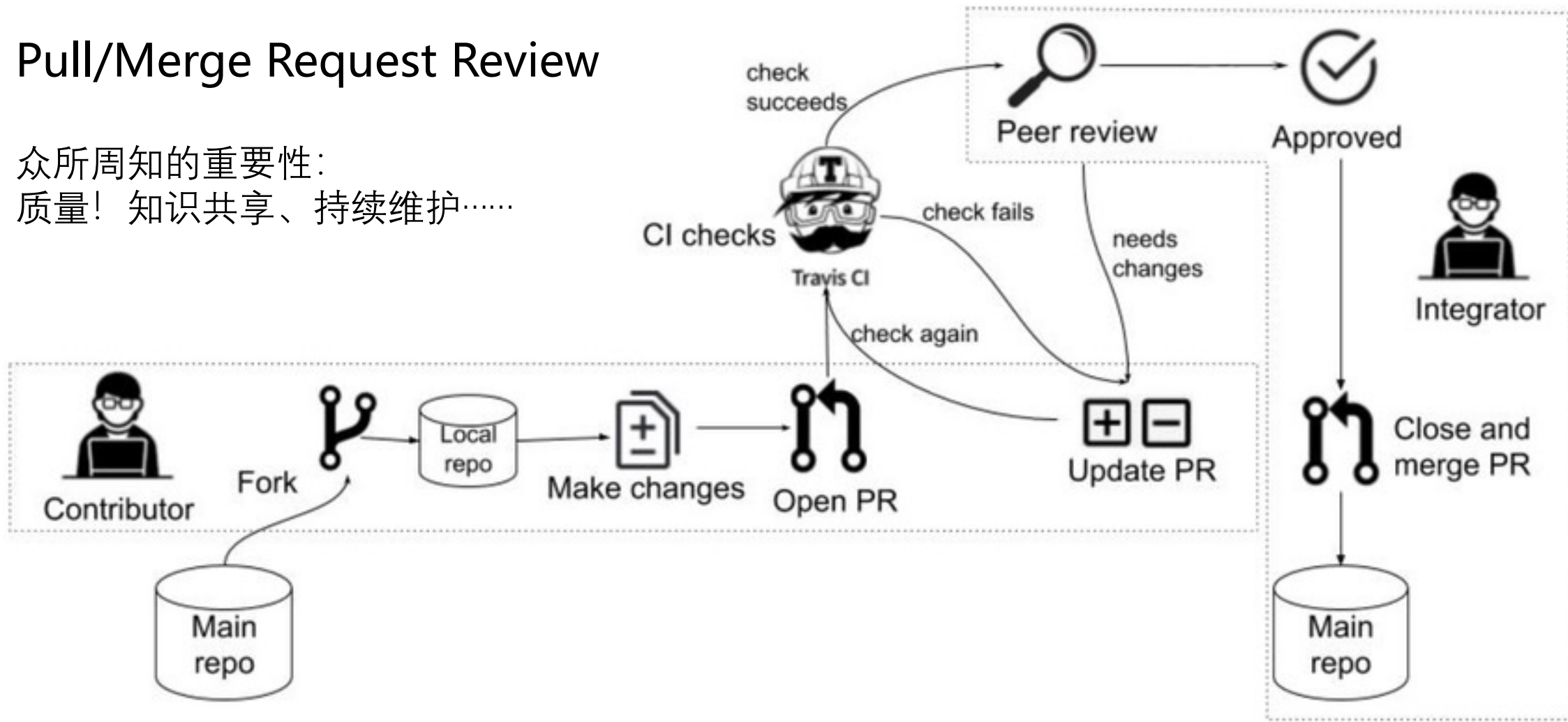
PART 01

背景与痛点 —— 让人又爱又恨的 Code Review 环节

▶ 背景与痛点 —— 让人又爱又恨的 Code Review 环节

Pull/Merge Request Review

众所周知的重要性：
质量！ 知识共享、持续维护……



▶ 背景与痛点 —— 让人又爱又恨的 Code Review 环节

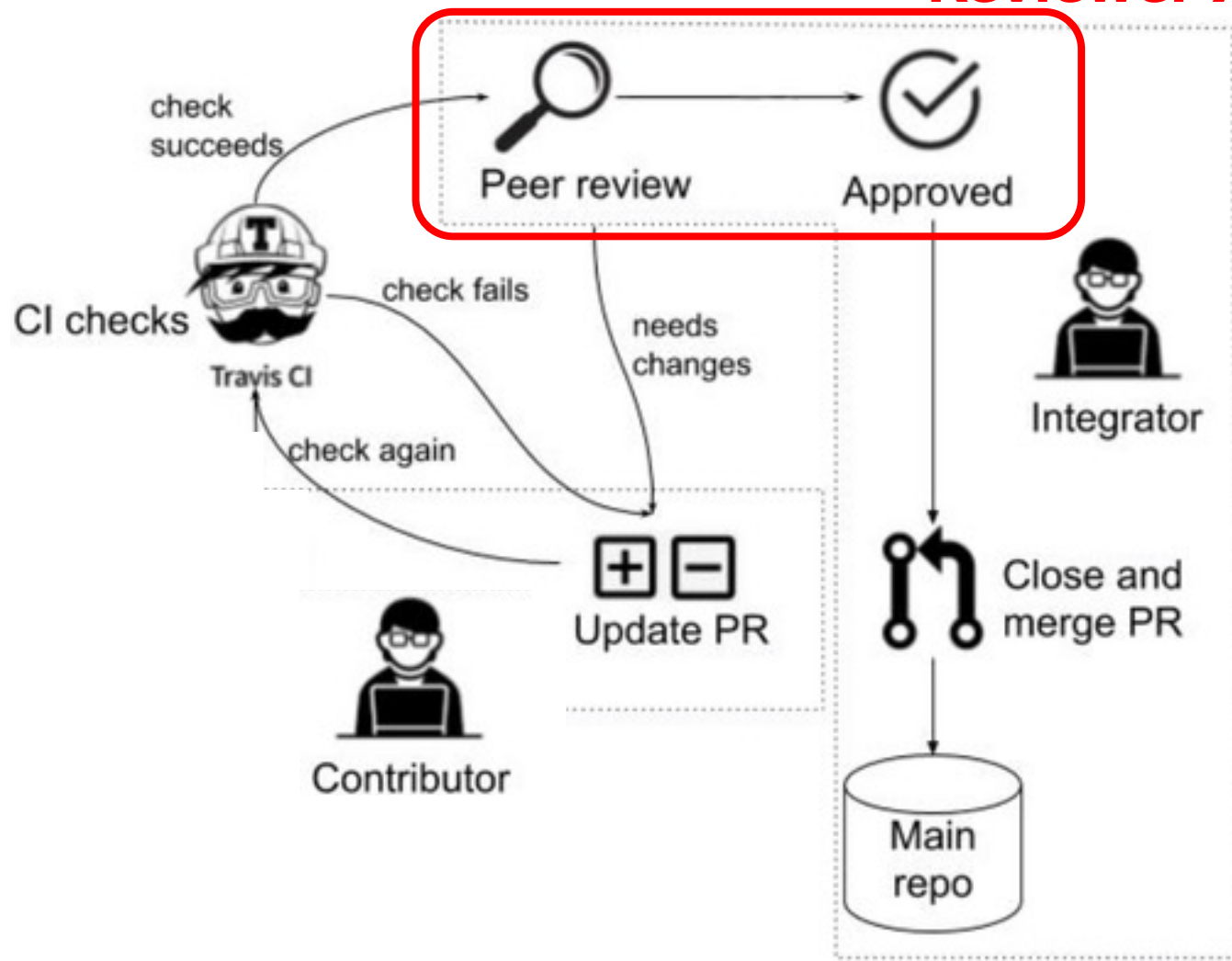
Reviewer X 2

Pull/Merge Request Review

众所周知的重要性：
质量！ 知识共享、持续维护……

但是，没有人喜欢 PR Review：

- 成本、短期效率
- 心理和社交压力
- 核心成员的精力分配与持续成长
-



▶ 背景与痛点 —— 让人又爱又恨的 Code Review 环节

LLM + Reviewer X 1

Pull/Merge Request Review

众所周知的重要性：
质量！知识共享、持续维护……

但是，没有人喜欢 PR Review：

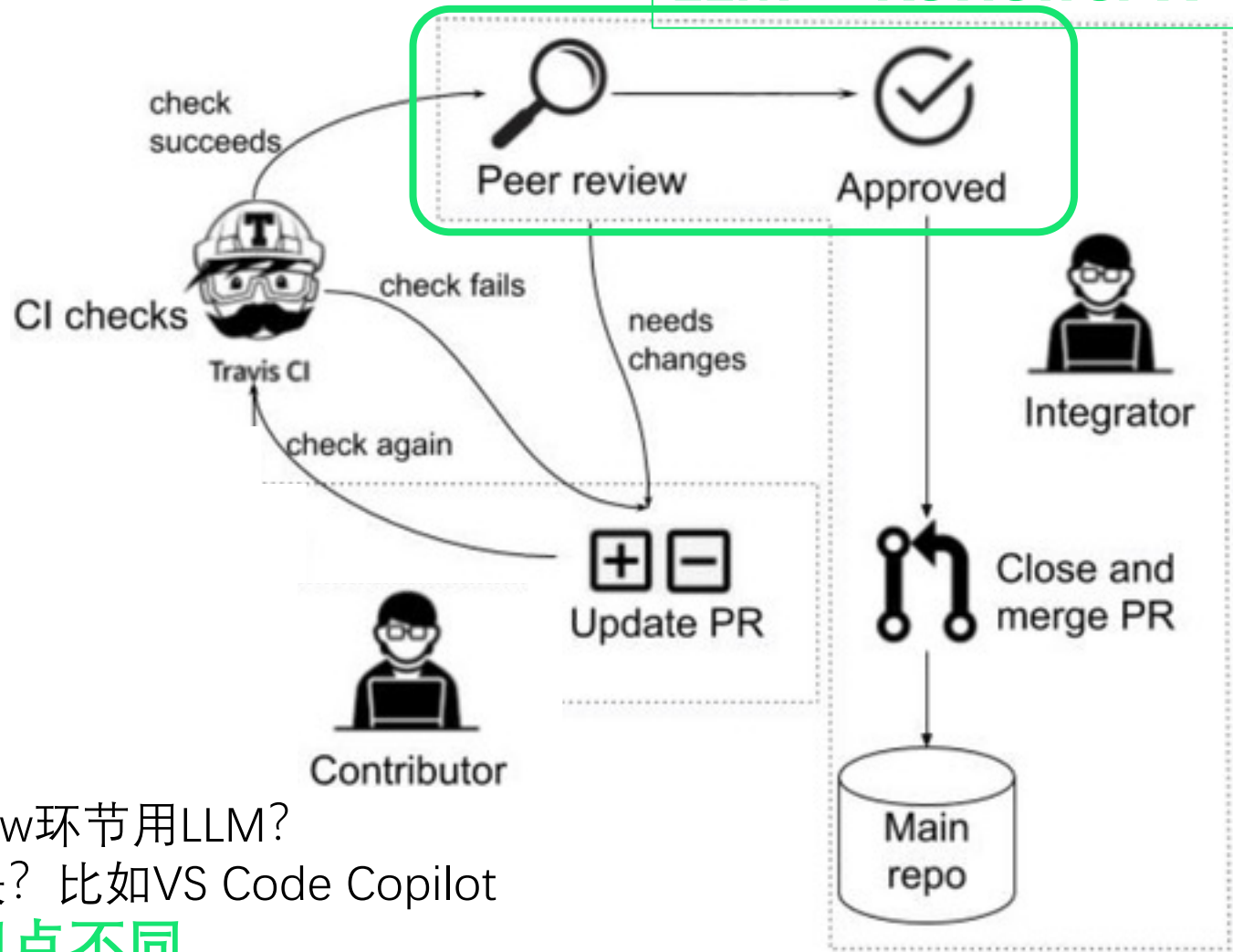
- 成本、短期效率
- 心理和社交压力
- 核心成员的精力分配与持续成长
-

另一个问题：

为什么要在 Pull/Merge Request Review 环节用 LLM？

为什么不在其他环节对代码审查中解决？比如 VS Code Copilot

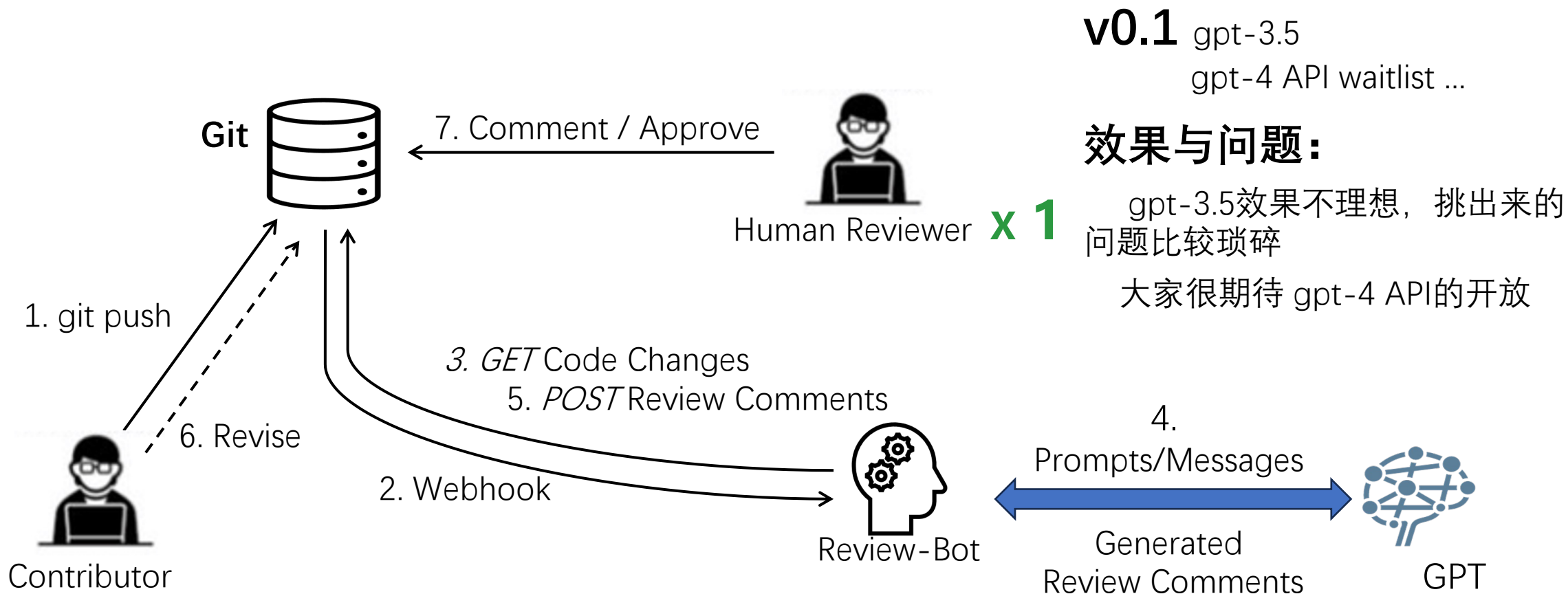
Review 对象不同，上下文、时间点不同



PART 02

Code-Chat-Review 开发历程 与实现原理

▶ Code-Chat-Review 开发历程与实现原理



```
messages = [  
  {  
    "role": "system",  
    "content":  
      """
```

As an AI assistant with expertise in programming, your primary task is to review the pull request provided by the user.

When generating your review, adhere to the following template:

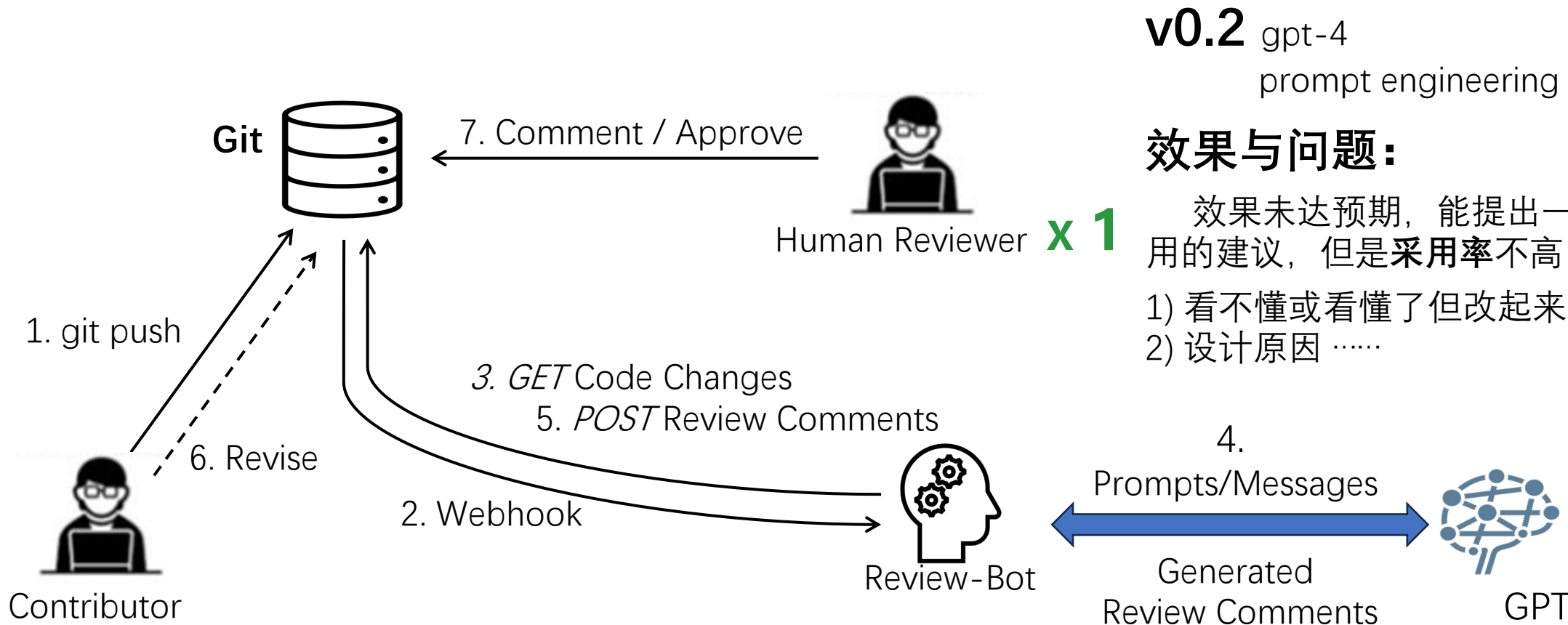
```
 **[Changes]**: Summarize the main changes made in the pull request in less than 50 words.  
 **[Suggestions]**: Provide any suggestions or improvements for the code. Focus on code quality, logic,  
 potential bugs and performance problems. Refrain from mentioning document-related suggestions such as "I  
 suggest adding some comments", etc.  
 **[Clarifications]**: (Optional) If there are parts of the pull request that are unclear or lack  
 sufficient context, ask for clarification here. If not, this section can be omitted.  
 **[Conclusion]**: Conclude the review with an overall assessment.  
 **[Other]**: (Optional) If there are additional observations or notes, mention them here. If not, this  
 section can be omitted.
```

The user may also engage in further discussions about the review. It is not necessary to use the template when discussing with the user.

```
      """  
    },  
    {  
      "role": "user",  
      "content": f"Review the following pull request. The patches are in standard `diff` format. Evaluate the  
 pull request within the context of the referenced issues and full content of the code  
 file(s).\n{changes_str}\n",  
    },  
  ]
```



Code-Chat-Review 开发历程与实现原理



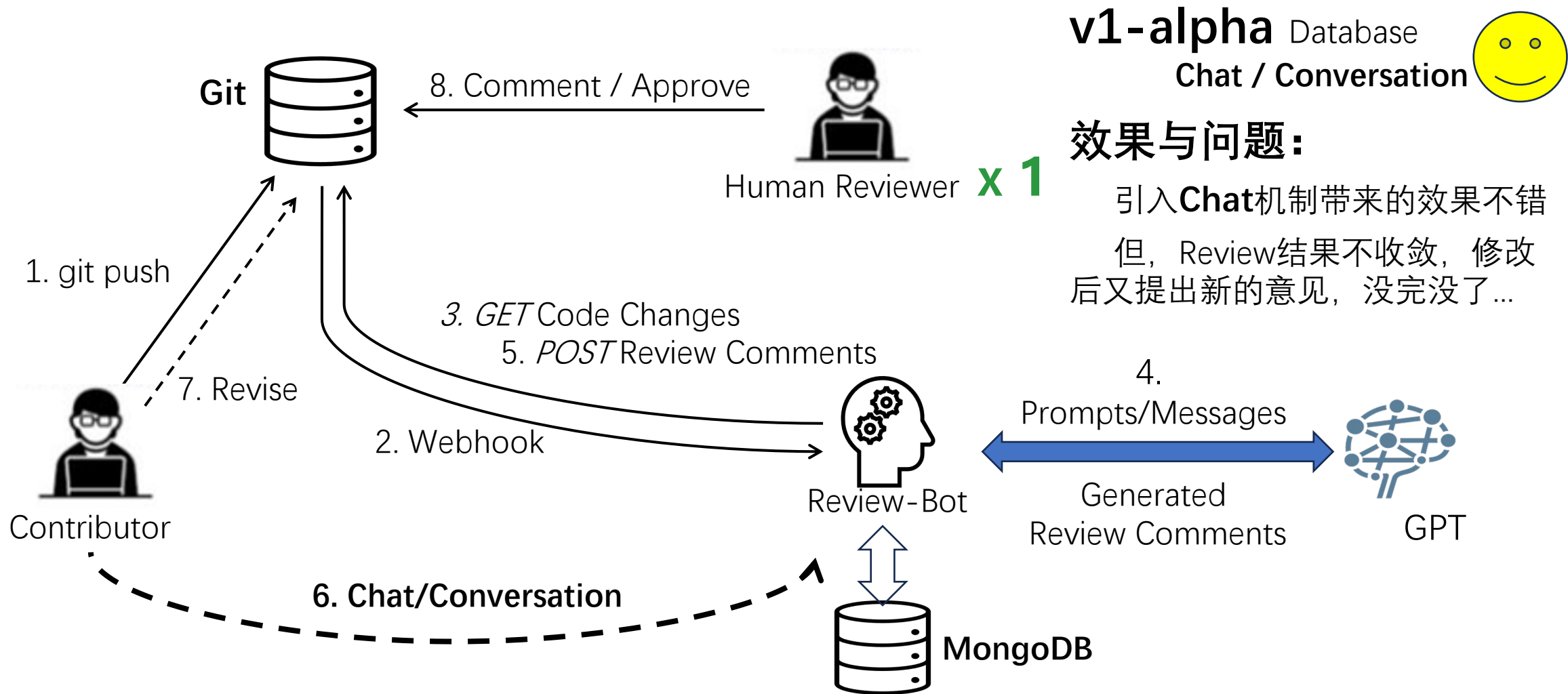
v0.2 gpt-4
prompt engineering 

效果与问题:

效果未达预期，能提出一些有用的建议，但是采用率不高:

- 1) 看不懂或看懂了但改起来费劲;
- 2) 设计原因

Code-Chat-Review 开发历程与实现原理



[AI Review] This comment is generated by an AI model (gpt-4) via v2 prompt.

[Changes]: The pull request introduces a Helm chart for deploying a Minecraft server on a Kubernetes cluster. It includes several Kubernetes resources such as Deployment, Service, PersistentVolumeClaim, and ConfigMap. The chart also includes a values file to customize the deployment.

[Suggestions]:

1. The `server.properties` in the ConfigMap is directly taken from `.Values.serverProperties`. This might expose sensitive data such as `rcon.password`. Consider using Kubernetes Secrets for sensitive data.
2. The `serverName` is directly used in resource names. This might cause issues if the server name contains characters that are not allowed in Kubernetes resource names. Consider sanitizing the server name before using it in resource names.
3. The Service is of type `NodePort` which might not be suitable for all environments. Consider making the Service type configurable through the values file.
4. The Deployment uses the `Recreate` strategy. This means downtime during updates. Consider if this is the best strategy for your use case.

[Clarifications]: None.

[Conclusion]: The pull request is a good start for deploying a Minecraft server using Helm. However, it needs some improvements to enhance security and flexibility.

[Other]: None.

[Note]

The above AI review results are for reference only, please rely on human expert review results for the final conclusion. Usually, AI is better at enhancing the quality of code snippets. However, it's essential for human experts to pay close attention to whether the modifications meet the overall requirements. Providing detailed information in the PR description helps the AI generate more specific and useful review results.

For further discussion with the AI Reviewer, please visit: <http://8.210.154.100:80705/.../4e2a-8a0f-028106feb972>

database conversation 

带来的效果不错
果不收敛，修改
1，没完没了...



GPT

1. git p



Contribu

Code-C

USER

下面我们讨论你在review comment中提到的第2点。You said "sanitizing the server name before using it in resource names." How to implement it when using Helm?

ASSISTANT

In Helm, you can sanitize the server name using template functions provided by the Go templating engine. One of the commonly used functions for sanitizing is `regexReplaceAll`.

For instance, if you want to replace any non-alphanumeric characters in the server name with a hyphen, you can use the following expression:

```
{{ .Values.ServerName | regexReplaceAll "[^a-zA-Z0-9]" "-" | trunc 63 | trimSuffix "-" }}
```

This will replace any character that is not a letter or a digit with a hyphen, truncate the result to 63 characters (the maximum length for many Kubernetes resources), and remove any trailing hyphens.

You can use this expression wherever you use the server name in the resource names. For example:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: mc-config-{{ .Values.ServerName | regexReplaceAll "[^a-zA-Z0-9]" "-" | trunc 63 | trimSuffix "-" }}
```

Please note that this is a simple example and you may need to adjust the regular expression and other transformations according to your needs.

USER

I don't want to replace any character without noticing the user. Just show a message when an invalid serverName is used.

ASSISTANT

In that case, you can use the `fail` function provided by Helm. This function allows you to stop the rendering process and return an error message. Here is an example:

```
{{- if not (regexMatch "^[a-zA-Z0-9-]*$" .Values.ServerName) }}
{{- fail "Invalid serverName. The serverName can only contain alphanumeric characters and hyphens." }}
{{- end }}
```

This will check if the `ServerName` only contains alphanumeric characters and hyphens. If it doesn't, Helm will stop rendering and return the error message "Invalid serverName. The serverName can only contain alphanumeric characters and hyphens."

use

sation



的效果不错

收敛，修改完没了...

1. git push

7.1

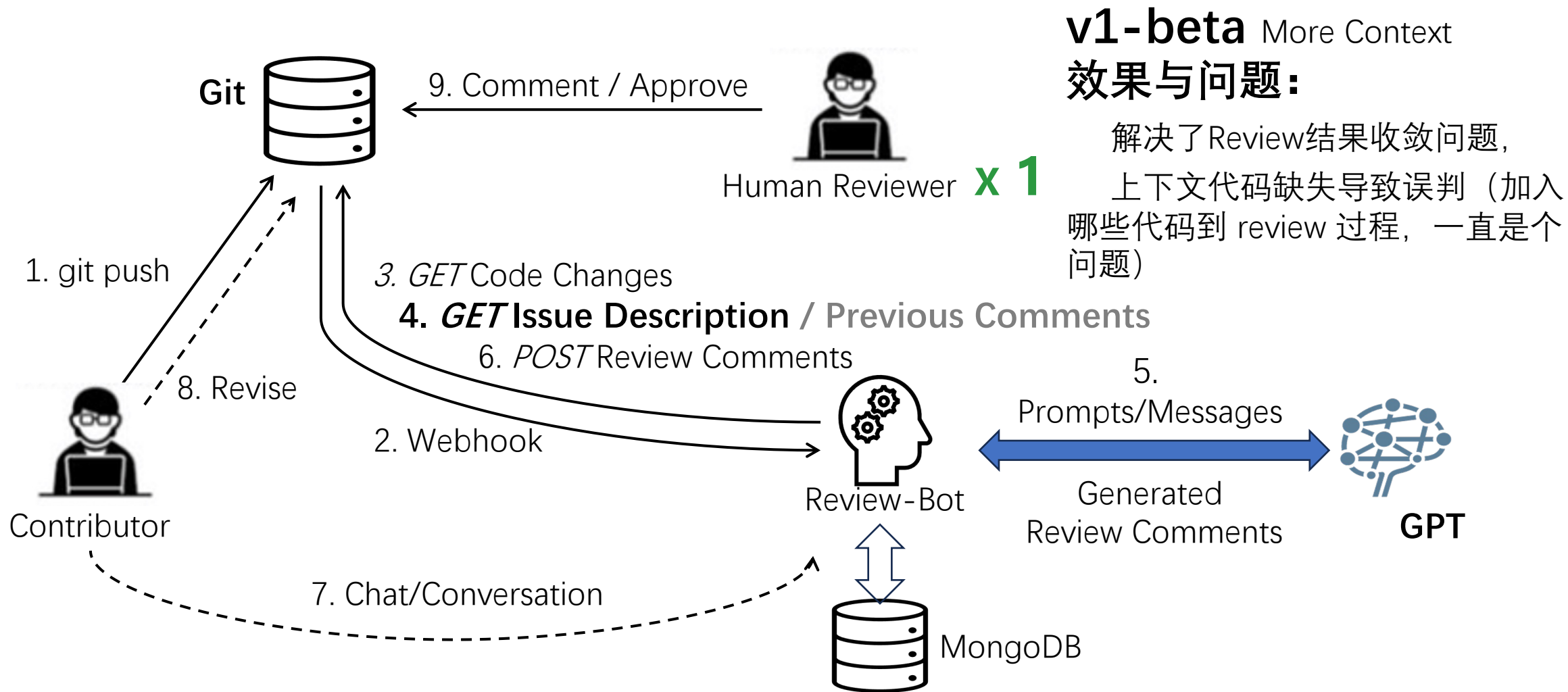


Contributor



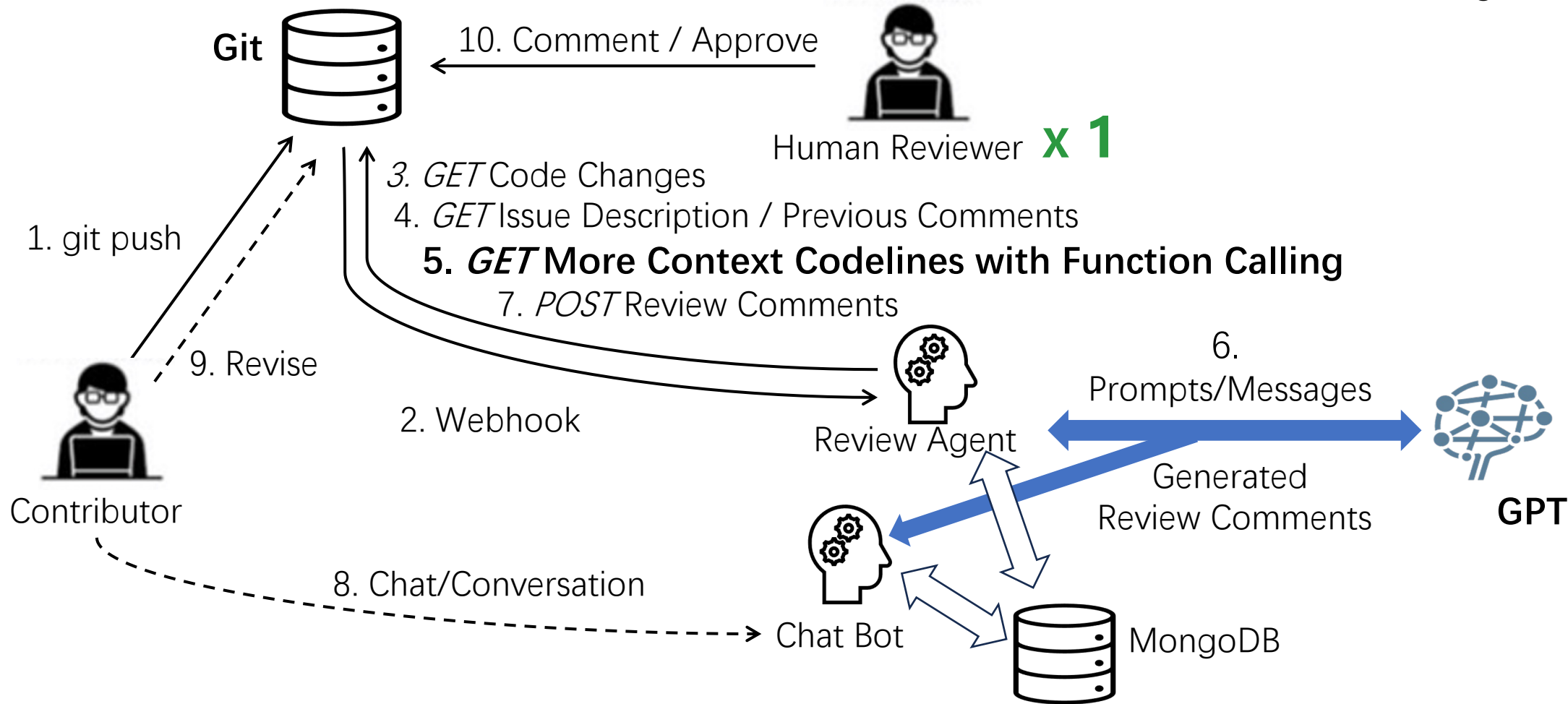
GPT

▶ Code-Chat-Review 开发历程与实现原理

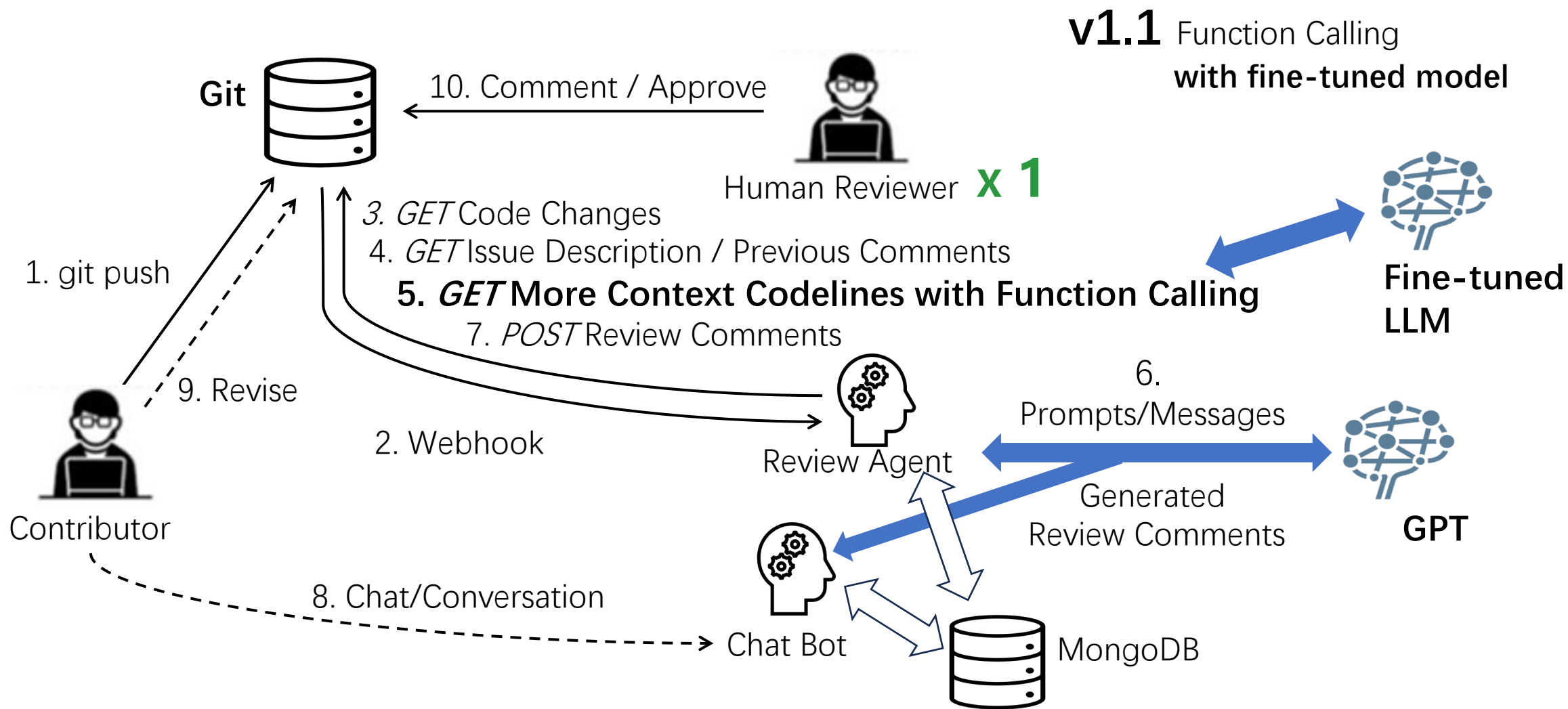


▶ Code-Chat-Review 开发历程与实现原理

v1.0 Function Calling



Code-Chat-Review 开发历程与实现原理



PART 03

应用效果与优缺点



▶ 应用效果 —— 优点

<https://github.com/OpenRHINO/RHINO-Operator/pull/51>

“为“等待Minikube”步骤添加超时，以防Minikube启动失败时 workflow 无限期地运行”

<https://github.com/LI-Mingyu/cndev-tutorial/pull/23>

“在Chart.yaml中，appVersion被设置为“1.16.0”。确保它与您要部署的实际应用程序版本匹配。”

<https://github.com/OpenRHINO/RHINO-Operator/pull/38>

“在Reconcile函数中使用常量而不是硬编码的字符串来表示pod和job的状态值”

应用效果 —— 优点

For example, if this pod is intended to interact with other services in the Kubernetes environment, it would be beneficial to see the service configuration and how the Python application inside the container is set up to discover and interact with these services. This could potentially lead to suggestions on improving the service interaction or handling potential failures in service discovery.

Additionally, if the container is expected to serve requests, it would be helpful to know how the service is exposed (e.g., via a Kubernetes Service or Ingress). This could potentially lead to suggestions on improving the service exposure or handling potential traffic issues.

USER

This is a prototype implementation. So don't provide suggestions on fault tolerance, safety, or security concerns, which are not addressed at this stage.

ASSISTANT

Understood. For a prototype implementation, your Kubernetes configuration is generally well-structured. Here are a few general suggestions that might help improve the code:

- Container command:** The current command (`sleep 100000`) essentially puts the container into a sleep state for a long duration. If the purpose of this is to keep the container running indefinitely for testing, you might consider using a command that makes this more explicit, such as `tail -f /dev/null`.
- Resource limits:** The resource limits are set quite low (`memory: "128Mi", cpu: "200m"`). If your application might need more resources during testing, you may want to increase these limits. Conversely, if you know your application requires less resources, lowering these limits can make your application more efficient.
- Image versioning:** You are using a specific version of the image (`istio/examples-bookinfo-productpage-v1:1.16.2`). This is generally good practice, as it ensures your testing environment is stable. However, make sure to update this version as needed to incorporate any updates from the image.
- ImagePullPolicy:** The `IfNotPresent` policy is efficient because it avoids pulling the image if it already exists on the node. However, if you are actively developing and pushing new versions of the image, you might want to use the `Always` policy to ensure you are always running the latest image.

Please keep in mind that these

Send

Any further suggestions related to my pull request?

This is a prototype implementation. So don't provide suggestions on fault tolerance, safety, or security concerns, which are not addressed at this stage.

针对生产和原型系统

提出不同的修改意见

应用效果 —— 优点

```
enableButtons();  
});  
}
```

对于第三条建议，你的观察是正确的，有

USER

刚才你的回答被截断了，请继续“对于第三条建议，你的观察是正确的，有”往下说

ASSISTANT

① 交流设计考量

对不起，我的回答确实被截断了。我想说的是，对于第三条建议，你的观察是正确的，有些用户可能会觉得弹出警告框比较烦人。如果用户只是误触了Send按钮，那么直接不响应可能就是最好的解决方案。但如果用户确实想要发送消息，但是他们没有意识到他们的输入只包含空白字符，那么一个友好的提示可能会很有帮助。

一个可能的解决方案是在文本输入框下方添加一个小的提示文本，当用户尝试发送只包含空白字符的消息时，这个提示文本会显示出来。这样的提示方式相比弹出警告框来说，可能会更加友好和不那么打断用户的操作流程。

USER

那这种提示方式如何实现呢？

ASSISTANT

② 给出实现代码

你可以通过在 HTML 中添加一个提示元素，然后在 JavaScript 中使用 DOM 操作来控制其显示和隐藏。下面是一个基本的实现方式：

首先，在 HTML 中添加一个提示元素：

```
<p id="inputError" style="color: red; display: none;">Please enter a valid message.</p>
```

然后在 JavaScript 函数中添加对这个提示元素的控制：

▶ 应用效果 —— 不足

- 一些简单的问题并没有检查出来（但大部分应该可以通过自动化测试避免）

```
print(reviews_str)
# Call GPT to generate the summary of the reviews
summary_messages = [
-   {"role": "system",
-   "content": f"Here are some review results for reference:\n{reviews_str}"},
  {"role": "user",
-   "content": "You are a software developing expert. Please summarize the review results. Ensure that the output follows the templ
  [Other]**\n\n' "}]
+   "content": "You are a software developing expert. Please summarize the review results:\n{reviews_str}\n\nEnsure that the output
  [Conclusion]**\n\n**[Clarifications]**\n\n**[Action]**\n\n**[Other]**\n\n'."}
]

summary_response = openai.ChatCompletion.create(
```

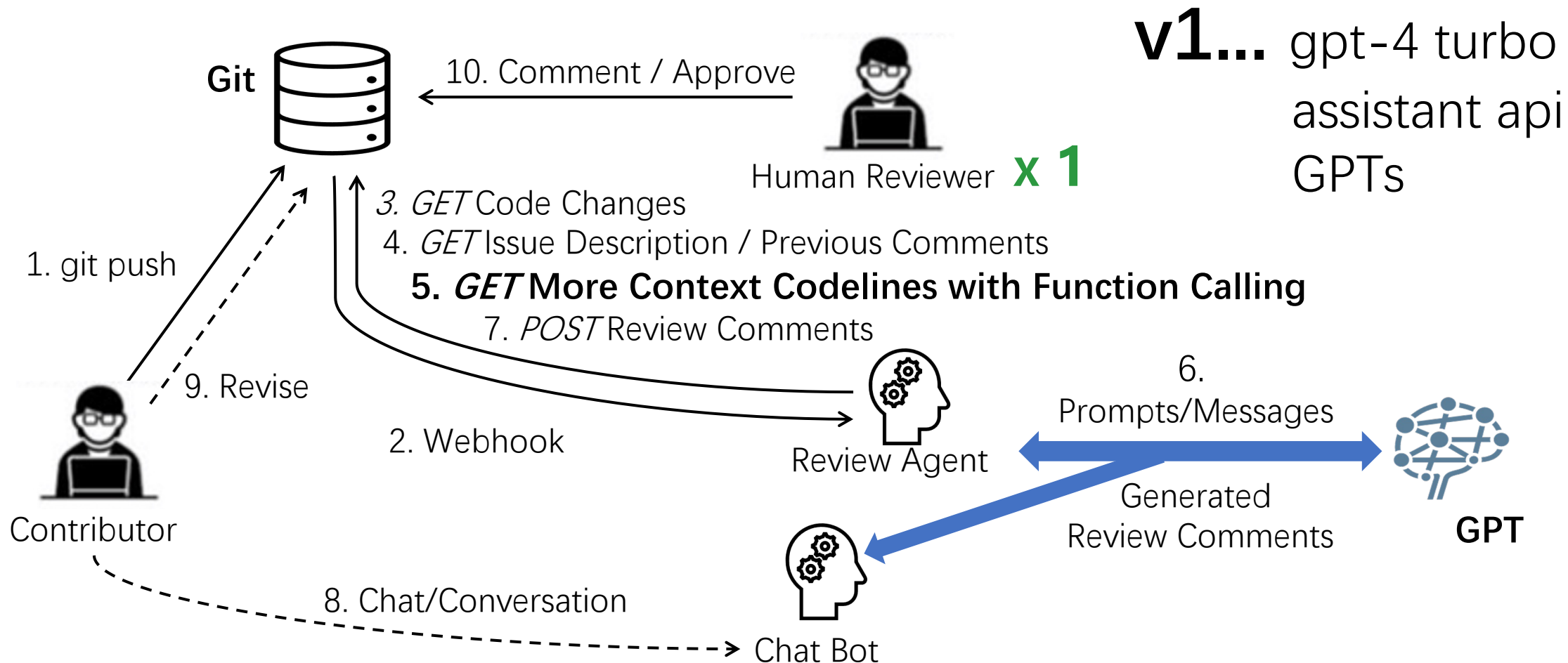
- 缺少对项目整体上下文的考虑
未来 100K 以上 tokens context-length 模型的测试
(一行代码大约10~20个tokens)

- 还是会提一些琐碎的问题
- 对自身代码Review时容易混淆 Prompts，对其他同类项目有可能存在类似问题？（未验证）

PART 04

11月6日以来的新进展

▶ 11月6日以来新进展



▶ GPT-4 Turbo

128K Tokens Context Length, 最够多的上下文, 但复杂大项目还是不行。Function Calling 获取上下文需要微调。

后续讨论仍然生硬地套格式, 虽然在最开始的 system prompt 里已经有了 "... It is not necessary to use the template when discussing with the user..."

解决方法 (很简单) 后续讨论组装 messages, 把 system prompt 排除在外

USER

我的代码中涉及到哪些 Flask application? 哪些 K8s yamls 又怎么指定端口的? 具体哪里有可能有问题?

ASSISTANT

[Changes]: Upon thorough review of the provided content, two Flask applications are affected: `conversation/conversation.py` and `pr_review/pr_review.py`. There are also two associated Kubernetes YAML files: `kubernetes/conversation_gpt.yaml` and `kubernetes/pr_review_gpt.yaml`.

[Suggestions]:

1. For the Flask application `conversation/conversation.py`, the `app.run()` method at the end of the full content does not specify a port, which means it defaults to port `5000`. However, in the `kubernetes/conversation_gpt.yaml`, the readiness probe is set to port `8080`, and the container exposes port `5000`.

To resolve this potential issue, you have two options:

- Modify the Flask app to run on port `8080` by adding the argument to the `app.run()` method: `app.run(host='0.0.0.0', port=8080)`.
- Update the Kubernetes YAML readiness probe to match the Flask default port `5000`.

2. For the Flask application `pr_review/pr_review.py`, the `app.run()` method at the end of the full content specifies the port as `8080`, which matches the Kubernetes YAML file readiness probe port in `kubernetes/pr_review_gpt.yaml`.
3. You also have consistency with your Kubernetes Service and Deployment YAML configurations. In the `kubernetes/conversation_gpt.yaml` file, the target port for the Service is correctly set to `5000`. In the `kubernetes/pr_review_gpt.yaml` file, the Service uses port `80` that routes to the target port `8080`, aligning with the Deployment's container port.

[Clarifications]:

- Ensure that the Flask applications are consistently configured to run on the ports intended to be exposed and probed in the Kubernetes configuration.

[Conclusion]: The `conversation_gpt` Flask application and Kubernetes YAML might have a port mismatch; ensure to align the ports correctly. The `pr_review_gpt` Flask application and the corresponding Kubernetes YAML are properly aligned regarding port configuration.

[Other]: None.

New GPT

Draft

Save

```

license:
  name: MIT
servers:
  - url: https://openrhino.org
paths:
  /pr_content:
    get:
      summary: Get content of a GitHub Pull Request
      operationId: getPRContent
      tags:
        - pull-request
      parameters:
        - name: repo_full_name
          in: query
          description: Full name of the repository (e.g., "owner/repo")

```

Available actions

Name	Method	Path	
getPRContent	GET	/pr_content	Test
getFileContent	GET	/file_content	Test
getIssueInfo	GET	/issue_info	Test

Authentication

None

Privacy policy

https://api.example-weather-app.com/privacy

Preview

- You**
 - Call the openrhino.org API with the getPRContent operation
 - ✓ Tested getPRContent
- Untitled**
 - > [debug] Calling HTTP endpoint
 - > [debug] Response received
 - > [debug] Calling HTTP endpoint
 - > [debug] Response received
 - ✓ Talked to openrhino.org

Message GPT...

上传、下载图片、
Code Interpreter

GPTs看不到构建历史，
也无法调试
用户访问本身无日志，
但是从外部调用可以
记录日志

RAG效果一言难尽.....
复杂场景目前还是需要
接入第三方服务

The screenshot displays the OpenAI ChatGPT interface. On the left, a sidebar shows the user's profile (Mingyu Li) and a list of GPTs, including 'RHINO Review Helper'. The main chat area shows a conversation with 'RHINO Review Helper'. The user asks: "Review this PR: <https://github.com/OpenRHINO/code-chat-reviewer/pull/37>". The GPT responds with a detailed review, including a list of changes and suggestions. The changes section states: "[Changes]: The pull request updates dependencies in `requirements.txt` for two modules: `conversation` and `pr_review`. The updates include setting Flask to version 3.0.0, pymongo to version 4.6.0, and openai to version 0.28. This aims to fix compatibility issues with openai versions $\geq 1.0.0$ and resolves issue #36." The suggestions section lists three items: 1. Version Consistency: Ensure all dependencies across different modules use consistent versions where applicable, to avoid potential compatibility issues. 2. Testing: Verify if the updated versions of Flask and pymongo introduce any breaking changes or require code modifications elsewhere in the project. 3. Dependency Management: Consider using a dependency management tool to... The interface also shows a search bar at the bottom and a footer note: "ChatGPT can make mistakes. Consider checking important information."

▶ Assistant API

Assistant API

不用保存对话历史了

很方便地换System Message
(Instruction)

目前不支持回调

Playground Assistants

Code Chat Reviewer

Name

Code Chat Reviewer

asst_8Q4gf87T2848CdaH5v8Ry7XA

Instructions

Your are an AI assistant with expertise in programming.

Model

gpt-4-1106-preview

TOOLS

Functions + Add

Code interpreter

Retrieval

FILES 📎 Add

Add files to use with code interpreter or retrieval.

THREAD thread_DAKrx9qcCMnGkMcMgWwD47C5

▶ Run

🗑 Clear

on the knowledge I have up to my last update.

User

访问kubernetes.io/docs, 然后告诉我告诉最新版本的k8s健康检查接口有没有变化?

Code Chat Reviewer

As an AI, I'm unable to directly access external websites, including the official Kubernetes documentation at kubernetes.io/docs, or search the internet in real-time. However, I can provide you with the conventional way to check this:

- 1 Visit the official Kubernetes documentation website (kubernetes.io/docs).
- 2 Navigate to the reference section, where you can find exhaustive details on the different API versions and their changes.
- 3 Look for the section relevant to `Pod` lifecycle or "Probes", usually found under "Concepts" -> "Containers".
- 4 Review the latest release notes, often found at kubernetes.io/releases/, which detail any changes, deprecations, or new features added in the recent Kubernetes versions.
- 5 Compare the probe configuration from the version you're

Enter your message...

▶ Add and run

Add

📎

PART 05

总结与展望



▶ 整体效果与体会

整体效果：

审核工作量**降低 70+ %**

局部问题导致的代码质量问题**减少 40+ %**

核心程序员代码贡献量**提高一倍**以上

低频场景，异步：

可以接受比较贵（常常也慢）的模型，

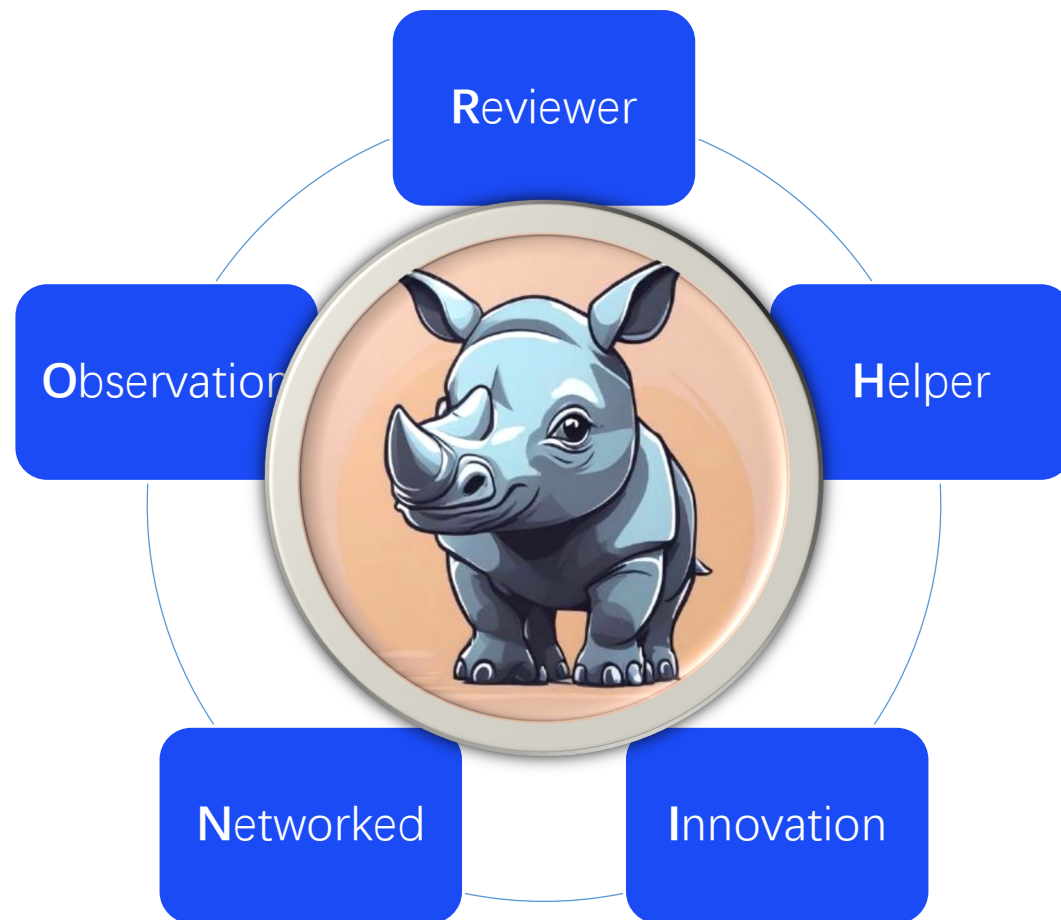
可以构造比较复杂的调用链以及多模型融合

▶ 未来计划

- 系统性地标准化测试
- 用其他大模型测试比较及融合（多个 Review Agent 印证与互补）
- 模型微调
- 通过向量数据库接入人类专家知识，改善对特定项目的Review效果
- 将代码审查与Git之外的信息（例如需求文档）结合

▶ 未来计划 RHINO, 基于大模型的科研计算平台

从论文idea → 实验代码 → 实验代码及结果Review → 论文辅助撰写/修改



THANKS

