

AutoCodeReview 配置指南

本文档将指导您如何配置 GitLab 连接和 AI 分析功能，实现自动化代码审查。

目录

- [GitLab 配置](#)
- [AI 配置 \(通过 OpenWebUI\)](#)
- [配置验证](#)
- [如何使用 AutoCodeReview 进行代码审查](#)
- [常见问题](#)

GitLab 配置

1. 获取 GitLab 访问令牌

步骤一：登录 GitLab

1. 登录到您的 GitLab 实例 (<http://gitlab.hzcctech.com>)
2. 点击右上角头像，选择 "Preferences"

步骤二：创建个人访问令牌

1. 在左侧导航栏中，点击 "Access Tokens/访问令牌"
2. 填写令牌信息：
 - **Token name:** `AutoCodeReview` (或任何描述性名称)
 - **Expiration date:** 设置合适的过期时间 (建议不超过1年)
 - **Select scopes:** 勾选以下权限：
 - `api` - 完整 API 访问权限
 - `read_repository` - 读取代码库
 - `write_repository` - 写入评论权限
3. 点击 "Create personal access token"
4. **重要：**立即复制生成的令牌，页面刷新后将无法再次查看

2. 在系统中配置 GitLab

访问配置页面

1. 登录 AutoCodeReview 系统
2. 点击右上角用户名，选择 "个人资料"
3. 在配置页面中填写：

GitLab URL

```
# 对于 gitlab.com
http://gitlab.hzcctech.com
```

访问令牌

```
# 粘贴刚才生成的个人访问令牌
glpat-xxxxxxxxxxxxxxxxxxxxxx
```

审查者名称

```
# 在 GitLab 评论中显示的名称（可选）
AutoCodeReview Bot
```

AI 配置（通过 OpenWebUI）

1. 在 OpenWebUI 中配置 AI 模型

获取 API 密钥

1. 在 OpenWebUI 中，进入 "Settings" → "Account"
2. 在 "API Keys" 部分，点击 "Create API Key"
3. 复制生成的 API 密钥

2. 在 AutoCodeReview 中配置 AI

AI API URL

```
# OpenWebUI 的 API 地址
http://ai.hzcctech.com/openai
```

AI API 密钥

```
# 在 OpenWebUI 中生成的 API 密钥
sk-xxxxxxxxxxxxxxxxxxxxxxxxxx
```

AI 模型

```
# 使用 OpenWebUI 中配置的模型名称，推荐：
qwen3-coder-480b-a35b
```

配置验证

1. GitLab 连接测试

在配置页面中：

1. 填写完 GitLab 相关配置后
2. 点击 "测试 GitLab 连接" 按钮

3. 系统会验证：

- GitLab URL 是否可达
- 访问令牌是否有效
- 必要权限是否具备

成功示例

GitLab连接测试成功

失败示例及解决方案

连接失败：401 Unauthorized

→ 检查访问令牌是否正确

连接失败：403 Forbidden

→ 检查令牌权限是否足够

连接失败：404 Not Found

→ 检查 GitLab URL 是否正确

2. AI 配置验证





通过界面测试**

1. 提交一个测试 MR URL 进行代码审查
2. 观察是否能正常分析代码并生成评论

如何使用 AutoCodeReview 进行代码审查

前置条件

在开始代码审查之前，请确保：

-  已完成 GitLab 配置并测试连接成功
-  已完成 AI 配置并能正常工作
-  拥有目标 GitLab 项目的访问权限
-  MR 处于可审查状态

步骤一：获取 MR URL

从 GitLab 获取 MR URL

1. 打开 GitLab 项目页面
2. 进入 "Merge requests" 页面
3. 找到需要审查的 MR，点击进入详情页
4. 复制浏览器地址栏中的完整 URL

MR URL 格式示例

```
https://gitlab.com/username/project/-/merge_requests/123
https://your-gitlab-domain.com/group/project/-/merge_requests/456
```

注意事项

- URL 必须包含完整的协议 (http:// 或 https://)
- 确保 MR 编号正确
- 可以直接复制浏览器地址栏的 URL

步骤二：启动代码审查

访问审查页面

1. 登录 AutoCodeReview 系统
2. 在首页找到 "代码审查" 区域
3. 在 "Merge Request URL" 输入框中粘贴 MR URL

URL 验证 (可选)

1. 粘贴 URL 后, 点击 "验证 URL 格式" 按钮
2. 系统会检查 URL 格式是否正确
3. 看到 "URL格式正确" 提示后继续

启动审查

1. 点击 "开始审查" 按钮
2. 系统开始后台处理, 页面会显示审查 ID
3. 记录这个审查 ID, 用于后续跟踪进度

步骤三：跟踪审查进度

实时进度监控

1. 系统会自动跳转到进度页面
2. 或者手动输入审查 ID 查看进度
3. 进度页面显示:
 - 当前状态 (准备中/分析中/生成评论中/完成)
 - 已处理文件数量
 - 发现的问题数量
 - 当前正在分析的文件

审查阶段说明

- **准备阶段**: 获取 MR 信息和文件变更列表
- **分析阶段**: 逐个分析修改的代码文件
- **生成评论阶段**: 为发现的问题生成评论文本
- **完成阶段**: 所有分析完成, 等待确认

步骤四：查看审查结果

审查完成后

1. 进度页面会显示 "审查完成"
2. 点击 "查看结果" 按钮
3. 或直接访问 `/review/{review_id}/result` 页面

结果页面内容

- **审查摘要：**
 - 总共分析的文件数
 - 发现的问题总数
 - 按严重程度分类的统计
- **MR 信息：**
 - 标题和作者
 - 源分支和目标分支
 - GitLab 链接

问题分类

- **安全风险：**潜在的安全漏洞
- **语法错误：**语法或逻辑错误
- **性能优化：**性能相关的建议
- **代码风格：**代码规范和风格问题
- **代码审查：**一般性改进建议

步骤五：审查和确认评论

查看待确认评论

1. 在结果页面点击 "查看待确认评论"
2. 系统列出所有待确认的评论
3. 每条评论显示：
 - 文件路径和行号
 - 问题类型和严重程度
 - 问题描述和建议
 - 代码上下文（按需加载）

单个评论确认

1. 点击 "查看代码上下文" 查看相关代码
2. 仔细阅读问题描述和建议
3. 选择操作：
 - **确认并发布：**将评论发布到 GitLab MR

- **拒绝**：忽略这个评论

批量评论确认

1. 勾选要确认的评论
2. 点击 "批量确认" 按钮
3. 系统会将选中的评论一次性发布到 GitLab

代码上下文功能

- 显示目标行及前后 5 行代码
- 突出显示存在问题的代码行
- 提供足够的上下文便于判断

步骤六：在 GitLab 中查看评论

返回 GitLab MR 页面

1. 点击结果页面中的 GitLab 链接
2. 或直接访问原 MR URL
3. 刷新页面查看新的评论

评论显示格式

****安全风险****：此处存在 SQL 注入风险

****建议****：使用参数化查询来防止 SQL 注入攻击

评论位置

- 行级评论：直接显示在对应的代码行
- 一般评论：显示在 MR 讨论区

完整工作流程示例

场景：审查一个包含数据库操作的 Python MR

1. 开发者提交 MR：
`https://gitlab.com/myproject/backend/-/merge_requests/45`
2. 审查者操作：
 - 登录 AutoCodeReview
 - 输入 MR URL
 - 点击"开始审查"
 - 获得审查 ID: 123
3. 系统处理：
 - 分析 3 个修改的 Python 文件
 - 发现 5 个问题（2个警告，3个建议）
 - 生成 5 条评论

4. 审查确认：

- 查看所有 5 条评论
- 确认其中 4 条有用的评论
- 拒绝 1 条误报评论
- 批量发布到 GitLab

5. GitLab 展示：

- MR 页面显示 4 条新评论
- 开发者收到通知
- 根据建议修改代码

最佳实践建议

审查时机

- MR 创建后，合并前进行审查
- 代码稳定后再审查，避免频繁变更
- 大型 MR 建议分批审查

评论确认原则

- 仔细阅读每条评论，理解问题描述
- 结合代码上下文判断评论的准确性
- 优先确认安全风险和语法错误类评论
- 对于风格类建议，结合团队规范决定

团队协作

- 指定专人负责评论确认
- 建立评论确认的标准流程
- 定期回顾和优化审查质量

配置示例

完整配置示例

GitLab 配置

GitLab URL: `https://gitlab.example.com`
访问令牌: `glpat-abc123def456ghi789`
审查者名称: `CodeReview Bot`

AI 配置

AI API URL: `http://localhost:3000/api/v1`
AI API 密钥: `sk-abc123def456ghi789jkl012mno345pqr678`
AI 模型: `gpt-4`

常见问题

Q1: GitLab 连接失败怎么办？

检查清单

- ☐ GitLab URL 是否正确（包含协议 http://）
- ☐ 访问令牌是否有效且未过期
- ☐ 令牌权限是否包含 `api`、`read_repository`、`write_repository`
- ☐ 网络是否可达（防火墙、代理设置）

Q2: AI 分析不工作怎么办？

检查清单

- ☐ OpenWebUI 是否正常运行
- ☐ AI API URL 是否正确
- ☐ API 密钥是否有效
- ☐ 模型名称是否在 OpenWebUI 中存在
- ☐ 网络连接是否正常

Q3: 如何更换 AI 模型？

1. 在 OpenWebUI 中添加新的模型
2. 在 AutoCodeReview 配置中更新模型名称
3. 重新测试功能

Q4: 支持哪些 AI 模型？

通过 OpenWebUI，支持：

- 兼容 OpenAI API 的模型，推荐
 - qwen3-coder-480b-a35b

Q5: 如何提高代码审查准确性？

建议

- 使用更强大的模型
- 确保代码库有良好的文档和注释
- 定期更新 AI 模型
- 针对特定语言或框架调整提示词
- 文件大小合理

安全注意事项

访问令牌安全

- 定期轮换 GitLab 访问令牌
- 使用最小权限原则
- 不要在代码或日志中暴露令牌

API 密钥安全

- 妥善保管 OpenWebUI API 密钥
- 定期检查 API 使用情况
- 发现异常立即更换密钥

技术支持

如果遇到配置问题，请：

1. 检查系统日志获取详细错误信息
2. 参考本文档的常见问题部分
3. 联系系统管理员或开发团队

更新时间: 2025年9月

版本: v1.0