团队项目 需要进一步理解的事情

第一部分：GitLab 核心权限体系

权限从低到高：

角色	权限说明	通俗理解
Guest	只能看问题和文档，不能碰代码	参观者
Reporter	能报告问题，但还是不能改代码	质检员
Developer	能写代码、创建分支、提交合并请求	工程师
Maintainer	能合并代码、管理分支、配置自动化流程	项目经理
Owner	能转让或删除整个项目	公司老板

实际应用：

- 普通开发者通常是 **Developer** 角色
- 技术负责人或架构师是 **Maintainer** 角色
- 项目创建者默认是 **Owner**

第二部分：GitLab 核心协作概念详解

1. Issue（问题/任务）

是什么： 项目中的任何一个工作任务，比如：

- 新功能开发
- Bug 修复
- 优化改进

使用流程：

```
1. 发现需要做的工作 → 创建 Issue
```

```
2. 填写详细描述（最好用模板）
3. 分配给具体负责人
4. 关联到对应的里程碑
```

2. Issue 模板

为什么要用模板?
避免每个人写 Issue 时格式混乱，遗漏重要信息。

示例：Bug 报告模板

```
## Bug 描述
[简单说明什么问题]

## 复现步骤
1. 第一步操作
2. 第二步操作
3. 预期结果 vs 实际结果

## 环境信息
- 操作系统:
- 浏览器:
- 版本号:

## 截图/日志
[附上相关证据]
```

3. Task（任务清单）

是什么： 把复杂的 Issue 拆分成多个小步骤

示例:

```
开发用户登录功能:
- [ ] 设计数据库表结构
- [ ] 编写登录接口
- [ ] 编写前端登录页面
- [ ] 编写单元测试
- [ ] 编写接口文档
```

4. Epic（史诗）

是什么： 一个非常大的功能模块，包含多个相关的 Issue

示例：

```
Epic：电商平台用户系统
├── Issue：用户注册功能
├── Issue：用户登录功能
├── Issue：密码找回功能
└── Issue：个人信息管理
```

5. Milestone（里程碑）

是什么： 项目的重要时间节点或版本目标

示例：

```
里程碑：V1.0 版本 - 2024年6月发布
├── 完成用户登录功能
├── 完成商品浏览功能
└── 完成购物车功能
```

6. Issue Boards（任务看板）

是什么： 可视化的任务管理面板，像真实的物理白板

典型列设置：

```
待处理 → 进行中 → 代码审查 → 测试中 → 已完成
```

使用方式：

- 每个任务是一个卡片
- 拖拽卡片到不同列来更新状态
- 所有团队成员都能看到整体进度

7. Wiki（项目文档）

是什么： 项目的知识库和文档中心

应该包含什么：

- 项目架构设计

- API 接口文档
- 部署安装指南
- 开发规范
- 新人入职指南

与 README 的区别：

- README：项目快速入门（给新成员看）
- Wiki：详细技术文档（给开发团队看）

8. Unit Tests（单元测试）

是什么： 针对代码最小单元的自动化测试

为什么要写：

- 保证代码质量
- 避免改坏现有功能
- 是自动化部署的基础

示例：

```
// 测试登录功能
test('用户输入正确密码应该登录成功', () => {
  const result = login('user', 'correct-password');
  expect(result.success).toBe(true);
});
```

第三部分：完整协作流程示例

让我们通过一个真实场景来理解这些概念如何协同工作：

场景："开发用户登录"功能

第1步：规划阶段

1. 创建 **Epic**："用户认证系统"
2. 创建 **Milestone**："V1.0 登录模块 - 6月30日完成"

第2步：任务拆解

1. 创建 **Issue**："开发用户登录功能"，指派负责人
2. 使用模板填写详细需求
3. 在 Issue 中创建 **Task** 清单：

> - 设计数据库表
> - 编写后端接口
> - 编写前端页面
> - 编写单元测试

第3步：开发执行

1. 从 Issue 页面创建新分支 `feature/user-login`
2. 开发代码，完成一个 Task 就打勾 ✅
3. 编写 **Unit Tests** 确保代码质量
4. 推送代码，创建 Merge Request

第4步：代码审查

1. 在 **Issue Boards** 中把卡片拖到"代码审查"列
2. Maintainer 审查代码，运行自动化测试
3. 通过后合并到主分支

第5步：知识沉淀

1. 在 **Wiki** 中更新接口文档
2. 记录部署注意事项

第6步：完成验收

1. 在 **Issue Boards** 中拖到"已完成"列
2. 检查 **Milestone** 进度
3. 准备下一个功能开发

第四部分：AI 工具辅助（Claude Code）

什么是 Claude Code?

一个 AI 编程助手，可以帮你：

- 理解代码

- 编写代码
- 管理 GitLab 项目

基本使用方法：

1. **安装插件**：在 VSCode 中安装 Claude Code 插件
2. **获取 Token**：在 GitLab 生成 Personal Access Token
3. **初始化**：执行 `/init` 创建 CLAUDE.md 文件
4. **开始协作**：AI 会基于 CLAUDE.md 理解你的项目上下文

能帮你做什么？

- 自动创建 Issue 和 Merge Request
- 分析代码质量
- 生成测试用例
- 协助代码审查