开发流程（团队协作版）

阶段 1：准备工作（首次开发时执行）

1. 克隆远程仓库到本地

- 目的：获取项目的完整代码副本
- 操作：bash

```bash
git clone https://github.com/公司名/项目名.git  # 克隆远程仓库
cd 项目名  # 进入项目文件夹
```

2. 基于最新主分支创建自己的开发分支

- 目的：确保你的分支从团队最新代码开始，减少后续冲突
- 操作：bash

```bash
git checkout main  # 切换到主分支（克隆后默认已在main）
git pull origin main  # 拉取远程main的最新代码（关键！同步团队最新成果）
git checkout -b feature/你的名字/功能名  # 创建并切换到自己的分支（如feature/zhangsan/login）
```

阶段 2：开发过程中（重复执行）

1. 在自己的分支中开发功能

- 目的：独立修改代码，不影响主分支和其他分支
- 操作：直接在本地编辑项目文件（如添加代码、修改功能）

2. 定期提交本地修改

- 目的：保存自己的开发进度，防止代码丢失
- 操作：bash

```bash
git add .  # 将所有修改加入暂存区（准备提交）
git commit -m "完成了XX功能：具体说明改了什么"  # 提交到本地仓库（备注要清晰）
```

3. 开发中途同步团队最新代码（推荐每天 1 次）

- 目的：提前合并同事已提交的代码，避免后期冲突堆积
- 操作：bash

```bash
# 步骤1：更新本地主分支（获取同事的最新代码）
git checkout main  # 切回主分支
git pull origin main  # 拉取远程main的最新内容

# 步骤2：把主分支的新代码合并到自己的开发分支
git checkout feature/你的名字/功能名  # 切回自己的分支
git merge main  # 合并主分支的最新代码

# 若出现冲突（Git提示"Automatic merge failed"）：
# 1. 打开冲突文件，删除冲突标记（<<<<<<<、=======、>>>>>>>）
# 2. 保留正确的代码，保存文件
# 3. 解决后提交：
git add .
git commit -m "同步主分支代码，解决了XX文件的冲突"
```

阶段 3：功能完成后（最终提交）

1. 将本地开发分支推送到远程仓库

- 目的：让团队看到你的代码，方便后续审查和合并

- 操作：bash

```bash
git push origin feature/你的名字/功能名  # 推送本地分支到远程（首次推送可能需要按提示加-u参数）
```

2. 在 GitHub 发起合并请求（PR）

- 目的：申请将你的代码合并到主分支，接受团队审查
- 操作（网页端）：
  ① 打开项目的 GitHub 页面 → 点击“Pull requests”→“New pull request”
  ② 左侧选 `main`（目标分支），右侧选你的分支 `feature/你的名字/功能名`（来源分支）
  ③ 填写标题（如“新增用户登录功能”）和修改说明，点击“Create pull request”

3. 配合审查并修改（如有需要）

- 目的：确保代码符合团队规范和质量要求
- 操作：
  ① 查看同事的审查意见，在本地修改代码
  ② 修改后提交并推送（自动更新 PR）：bash

```bash
git add .
git commit -m "根据审查意见：修复了XX问题"
git push origin feature/你的名字/功能名
```

4. 代码通过审查后合并到主分支

- 目的：将你的功能正式合并到项目主线
- 操作（网页端）：
  审查通过后，在 PR 页面点击“Merge pull request”→“Confirm merge”

5. 合并后清理分支（可选但推荐）

- 目的：保持仓库整洁，避免分支过多混乱

- 操作：bash

```bash
# 更新本地主分支（获取合并后的最新代码）
git checkout main
git pull origin main

# 删除本地开发分支（功能已合并，无需保留）
git branch -d feature/你的名字/功能名

# 删除远程开发分支（可选，由团队统一决定）
git push origin --delete feature/你的名字/功能名
```

流程核心原则

- **隔离开发**：所有修改都在自己的分支进行，绝不直接改 `main`
- **频繁同步**：开发周期超过 1 天，每天同步 1 次主分支代码
- **清晰提交**：每次提交的备注要具体（避免“改了点东西”这种模糊描述）
- **审查优先**：代码必须通过 PR 审查才能合并，保证项目质量