Git & GitHub 新手学习指南 🚀

欢迎来到 Git 和 GitHub 的世界！这个指南将帮助你掌握版本控制的基础知识。

📚 目录

什么是 Git 和 GitHub

Git

- **Git** 是一个分布式版本控制系统
- 帮你追踪代码的变化历史
- 可以在不同版本间切换
- 支持多人协作开发

GitHub

- **GitHub** 是基于 Git 的代码托管平台
- 提供代码存储、协作、项目管理功能
- 全球最大的开源代码社区

Git 基础概念

📁 仓库 (Repository)

存储项目文件和版本历史的地方

📝 提交 (Commit)

保存项目某个时间点的快照

🌿 分支 (Branch)

独立的开发线，默认分支通常叫 `main` 或 `master`

🔄 合并 (Merge)

将一个分支的更改合并到另一个分支

常用 Git 命令

初始化和克隆

```
# 初始化新的 Git 仓库
git init

# 克隆远程仓库到本地
git clone https://github.com/username/repository.git

# 查看当前仓库状态
git status
```

添加和提交更改

```
# 添加单个文件到暂存区
git add filename.txt

# 添加所有更改的文件
git add .

# 提交更改并添加描述信息
git commit -m "你的提交信息"

# 一步完成添加和提交（只对已追踪的文件）
git commit -am "提交信息"
```

查看历史和差异

```
# 查看提交历史
git log

# 查看简洁的提交历史
git log --oneline

# 查看文件的更改
git diff

# 查看暂存区的更改
git diff --staged
```

分支操作

```
# 查看所有分支
git branch

# 创建新分支
git branch new-feature

# 切换到分支
git checkout new-feature

# 创建并切换到新分支（一步完成）
git checkout -b new-feature

# 合并分支到当前分支
git merge new-feature

# 删除分支
git branch -d new-feature
```

远程仓库操作

```
# 添加远程仓库
git remote add origin https://github.com/username/repository.git

# 查看远程仓库
git remote -v

# 推送到远程仓库
git push origin main

# 从远程仓库拉取更新
git pull origin main

# 获取远程更新但不合并
git fetch origin
```

GitHub 基础操作

1. 创建仓库

1. 登录 GitHub
2. 点击右上角的 "+" 按钮
3. 选择 "New repository"
4. 填写仓库名称和描述
5. 选择公开或私有
6. 点击 "Create repository"

2. 连接本地仓库到 GitHub

```
# 方法1：从现有本地项目
git remote add origin https://github.com/username/repository.git
git branch -M main
git push -u origin main

# 方法2：克隆 GitHub 仓库
git clone https://github.com/username/repository.git
```

3. Fork 和 Pull Request

- **Fork**：复制别人的仓库到你的账户
- **Pull Request**：请求将你的更改合并到原仓库

工作流程示例

日常开发流程

```
# 1. 克隆项目（只需要一次）
git clone https://github.com/username/project.git
cd project

# 2. 创建新分支进行开发
git checkout -b feature/new-feature

# 3. 进行开发工作
# ... 编写代码 ...

# 4. 查看更改
git status
git diff

# 5. 添加更改到暂存区
git add .

# 6. 提交更改
git commit -m "feat: add new feature"

# 7. 推送分支到 GitHub
git push origin feature/new-feature

# 8. 在 GitHub 上创建 Pull Request

# 9. 合并后切换回主分支并更新
git checkout main
git pull origin main

# 10. 删除已合并的分支
git branch -d feature/new-feature
```

团队协作流程

```
# 1. 每天开始工作前更新主分支
git checkout main
git pull origin main

# 2. 创建新的功能分支
git checkout -b feature/user-authentication

# 3. 开发并提交
git add .
git commit -m "feat: implement user login"

# 4. 定期推送（防止丢失工作）
git push origin feature/user-authentication

# 5. 完成开发后创建 Pull Request
# 6. 代码审查和合并
# 7. 清理本地分支
```

最佳实践

📝 提交信息规范

使用约定式提交格式：

```
# 格式：<类型>：<描述>
feat：添加新功能
fix：修复bug
docs：更新文档
style：代码格式调整
refactor：重构代码
test：添加测试
chore：其他杂项更改

# 示例
git commit -m "feat: add user registration functionality"
git commit -m "fix: resolve login button not working"
git commit -m "docs: update README with setup instructions"
```

🌿 分支命名规范

```
# 功能分支
feature/user-profile
feature/shopping-cart

# 修复分支
fix/login-error
hotfix/critical-security-issue

# 发布分支
release/v1.2.0
```

📋 .gitignore 文件

创建 `.gitignore` 文件来忽略不需要版本控制的文件：

```
# 依赖文件夹
node_modules/

# 环境变量文件
.env
.env.local

# 构建输出
.next/
dist/
build/

# 系统文件
.DS_Store
Thumbs.db

# IDE 文件
.vscode/
.idea/
```

常见问题解决

🚨 撤销更改

```
# 撤销工作区的更改（未添加到暂存区）
git checkout -- filename.txt

# 撤销暂存区的更改（已 add 但未 commit）
git reset HEAD filename.txt

# 撤销最后一次提交（保留更改）
git reset --soft HEAD^

# 完全撤销最后一次提交（删除更改）
git reset --hard HEAD^
```

🔧 修改提交信息

```
# 修改最后一次提交的信息
git commit --amend -m "新的提交信息"

# 修改最后一次提交并添加新文件
git add forgotten-file.txt
git commit --amend --no-edit
```

🔀 解决合并冲突

当两个分支修改了同一个文件的同一部分时会出现冲突：

1. Git 会标记冲突的文件
2. 手动编辑文件解决冲突
3. 添加解决后的文件： `git add conflicted-file.txt`
4. 完成合并： `git commit`

📦 储藏工作

```
# 临时保存当前工作
git stash

# 查看储藏列表
git stash list

# 恢复最新的储藏
git stash pop

# 恢复指定的储藏
git stash apply stash@{0}
```

🎯 练习建议

1. **每天练习基础命令**： `git status` ， `git add` ， `git commit`
2. **尝试分支操作**：创建、切换、合并分支
3. **模拟团队协作**：创建 Pull Request，进行代码审查
4. **学习撤销操作**：练习各种撤销场景
5. **阅读他人代码**：浏览 GitHub 上的开源项目

📚 推荐资源

- Git 官方文档
- GitHub 官方指南
- Learn Git Branching - 交互式学习
- Oh Shit, Git!?! - 解决 Git 问题

🎉 总结

Git 和 GitHub 是现代开发必备技能：

- **Git** 帮你管理代码版本和历史
- **GitHub** 提供协作和分享平台
- **多练习** 是掌握的关键
- **不要害怕犯错** - Git 可以恢复大部分操作

记住：每个开发者都是从新手开始的，持续练习你就能熟练掌握这些工具！

这个指南将随着你的学习进度持续更新。有问题随时查阅或寻求帮助！ 🚀