第三章：Git与代码托管平台入门

从本地版本控制到全球“社交化编程”

本章学习目标

通过本章学习，学生应能够：

- **理解版本控制**：掌握版本控制的基本概念、重要性和发展历程
- **掌握Git核心**：理解Git的核心理论、工作流程和基本操作命令
- **实践GitCode流程**：在GitCode上完成从创建仓库到提交PR的完整流程
- **体验开源协作**：初步体验在开源社区中通过Issue与他人协作交流

什么是版本控制 (VC)?

VC 是什么?

一个系统地记录一个或多个文件内容变化，以便将来查阅特定版本修订情况的系统。简而言之，它是项目的“历史记录仪”。

为什么至关重要?

- **团队协作:** 多人同时工作不冲突。
- **历史追溯:** 查看谁、何时、为何做了修改。
- **高效调试:** 快速定位引入Bug的变更。
- **安全网:** 随时可以“回滚”到任一历史版本。

版本控制演进历程

1. **第一代：本地版本控制系统 (LVCS)**
   1970s-1980s | **代表工具：** SCCS, RCS
   单机工具，缺乏网络功能，无法满足团队协作需求
2. **第二代：集中式版本控制系统 (CVCS)**
   1990s-2000s | **代表工具：** CVS, Subversion (SVN)
   引入中央服务器，解决协作问题，但存在单点故障风险
3. **第三代：分布式版本控制系统 (DVCS)**
   2005年至今 | **代表工具：** Git, Mercurial
   每个开发者拥有完整仓库副本，支持离线工作，分支管理强大

   Git的起源: 2005年，Linux创始人Linus Torvalds因原有的商业VCS (BitKeeper) 收回授权，在危机中仅用约两周时间创造了Git，以满足Linux内核开发的庞大需求。

Git 的核心工作流：三大区域

Git 的精髓在于其“暂存区”的设计，它让你能精确控制每一次“提交”的内容。

1. 工作区 你实际看到和编辑文件的目录。

`git add` →

2. 暂存区 “购物车”，存放下次提交的快照。

`git commit` →

3. 本地仓库 项目的完整历史数据库。

分支管理：Git 的超能力

什么是分支?

一个轻量级的“可移动指针”。它允许你在不影响主线 (`main`) 的情况下，安全地进行并行开发和实验。

基本流程

1. **创建并切换:** `git checkout -b new-feature`
2. **工作:** 编辑、`add`、`commit`...
3. **切回主线:** `git checkout main`
4. **合并:** `git merge new-feature`

图：Git分支管理逻辑示意图

远程协作：与团队同步

Git 是分布式的，代码托管平台 (如 GitCode) 是团队代码的“交汇点”。

⬇️ 拉取 (Pull)

`git pull`

从远程仓库获取最新更新，并自动合并到你的本地工作区。(相当于 `fetch` + `merge`)

⬆️ 推送 (Push)

`git push`

将你的本地提交 (commits) 上传到远程仓库，与团队分享你的工作成果。

“社交化编程”：Pull Request (PR) 完整流程

这是开源社区和现代团队协作的核心。你不能直接修改别人的项目，而是通过“拉取请求”来贡献你的代码。

1. **Fork (复刻)**
   在 GitCode 平台上，创建一份你自己的项目副本。
2. **Clone (克隆)**
   将**你自己的 Fork** 下载到本地电脑。
3. **Branch (创建分支)**
   在本地创建新分支 (`git checkout -b ...`) 来进行修改。
4. **Work (工作)**
   修改代码，然后 `git add` 和 `git commit`。
5. **Push (推送)**
   将你的新分支推送到**你自己的 Fork** (`git push origin ...`)。
6. **Create PR (创建拉取请求)**
   在 GitCode 网页上，请求原始项目“拉取”你的新分支。
7. **Review & Merge (审查与合并)**
   项目维护者审查你的代码，与你讨论，最终将其合并到主项目！

开源协作流程

1. Fork项目
创建项目的个人副本
获得完全控制权，可自由实验

2. 克隆到本地
`git clone` 你的Fork仓库
设置`upstream`跟踪原始项目

3. 创建分支开发
`git checkout -b` 新分支
在独立环境中工作，不影响主分支

4. 提交并推送
`git add, commit, push`
将更改推送到你的Fork仓库

5. 创建Pull Request
请求原始项目合并你的更改
进行代码审查和讨论

6. 合并到主项目
维护者审查后合并
你的贡献成为项目的一部分

> *"在数字协作的时代，Git不仅仅是一种工具，更是一种思维方式。"*