Kỹ năng chuyên môn: Quản lý Phiên bản với Git & GitHub

Thien Huynh-The

Bộ môn Kỹ thuật Máy tính
HCMC University of Technology and Education

Ngày 29 tháng 9 năm 2025

Tại sao cần học Git & GitHub?

Đây là những công cụ nền tảng trong ngành công nghiệp hiện đại:

- **Làm việc nhóm hiệu quả:** Cho phép nhiều người cùng đóng góp vào một dự án (code, tài liệu, thiết kế...) một cách có tổ chức, tránh xung đột.
- **Theo dõi và quay lui các thay đổi:** Dễ dàng xem lại lịch sử chỉnh sửa, biết được "ai đã thay đổi gì, khi nào", và có thể quay lại bất kỳ phiên bản nào nếu gặp lỗi.
- **Xây dựng "CV" chuyên nghiệp:** Một tài khoản GitHub với các dự án cá nhân và đóng góp thể hiện năng lực và sự đam mê của bạn với nhà tuyển dụng.
- **Lưu trữ an toàn và phân tán:** Mã nguồn của bạn được lưu trữ cả ở máy cá nhân và trên server. Nếu máy tính gặp sự cố, bạn không bị mất sản phẩm.
- **Nền tảng của quy trình phát triển hiện đại:** Là bước khởi đầu cho các quy trình tự động hóa chuyên nghiệp như CI/CD (Tích hợp/Triển khai liên tục).

Git & GitHub là gì?

Dù thường đi chung, chúng là hai thứ khác nhau:

Git

- Là một **phần mềm** (hệ thống quản lý phiên bản phân tán).
- Được cài đặt và chạy trên **máy tính cá nhân** của bạn.
- Nhiệm vụ: Theo dõi sự thay đổi của các file trong một thư mục dự án.
- Hoạt động không cần Internet.

GitHub (hoặc GitLab, Bitbucket)

- Là một **dịch vụ, nền tảng web**.
- Cung cấp nơi **lưu trữ trực tuyến** các dự án Git của bạn.
- Cung cấp các công cụ hỗ trợ làm việc nhóm: quản lý lỗi (issues), pull requests, thảo luận...
- Cần có Internet để tương tác.

Tóm lại: Bạn dùng Git *để quản lý dự án trên máy, và dùng* GitHub *để lưu trữ, chia sẻ dự án đó với người khác.*

Các khái niệm cốt lõi (Phần 1)

- **Repository (Repo - Kho chứa):** Là thư mục chứa toàn bộ dự án của bạn (bao gồm tất cả các file mã nguồn, hình ảnh, tài liệu...) cùng với lịch sử thay đổi của chúng. Có 2 loại:
  - **Local Repo:** Kho chứa trên máy tính của bạn.
  - **Remote Repo:** Kho chứa trên GitHub.

- **Commit (Bản ghi):** Giống như một "điểm lưu game". Mỗi khi bạn hoàn thành một phần công việc có ý nghĩa, bạn tạo một *commit* để ghi lại các thay đổi đó.
  - Mỗi commit có một mã định danh duy nhất và một thông điệp mô tả những gì đã được thay đổi.
  - Tập hợp các commit tạo thành lịch sử của dự án.

Các khái niệm cốt lõi (Phần 2)

- **Branch (Nhánh):** Là một "dòng thời gian" độc lập của các commit.
  - Nhánh mặc định thường tên là `main` hoặc `master`, chứa phiên bản ổn định nhất của sản phẩm.
  - Khi muốn phát triển một tính năng mới hoặc sửa một lỗi, bạn nên **tạo một nhánh mới** từ nhánh `main`. Mọi thay đổi trên nhánh mới này sẽ không ảnh hưởng đến nhánh chính.

- **Merge (Trộn/Gộp nhánh):** Là hành động lấy các thay đổi từ một nhánh và áp dụng chúng vào một nhánh khác.
  - Ví dụ: Sau khi hoàn thành tính năng trên nhánh mới, bạn sẽ *merge* nhánh đó trở lại vào nhánh `main` để cập nhật sản phẩm.

Luồng làm việc cơ bản

Đây là quy trình phổ biến nhất khi làm việc một mình:

1. **Clone:** Sao chép một remote repo (trên GitHub) về máy tính để tạo thành một local repo. (Chỉ làm 1 lần lúc bắt đầu).
2. **Add & Commit:** Sau khi chỉnh sửa code trên máy, bạn cần:
   - `git add`: Chọn những file thay đổi mà bạn muốn lưu.
   - `git commit`: Tạo một "điểm lưu" (commit) cho những thay đổi đó.
3. **Push:** Đẩy những commit từ local repo của bạn lên remote repo trên GitHub để đồng bộ.
4. **Pull:** Kéo những thay đổi mới nhất từ remote repo về local repo (hữu ích khi làm việc nhóm hoặc làm trên nhiều máy).

Bài tập thực hành tại lớp

Mục tiêu: Làm quen với luồng làm việc cơ bản: tạo repo, clone, commit, và push.
Yêu cầu chuẩn bị:

- **Tài khoản GitHub:** Đã đăng ký tài khoản tại `https://github.com/`.
- **Cài đặt Git:** Đã cài đặt Git trên máy tính.
  - Truy cập `https://git-scm.com/downloads` để tải về và cài đặt.
  - Sau khi cài, mở Terminal (macOS/Linux) hoặc Git Bash (Windows) và chạy 2 lệnh sau để cấu hình tên và email của bạn:

```
git config --global user.name "Your Name" \\
git config --global user.email "your.email@example.com"
```

Bước 1: Tạo Kho chứa (Repository) trên GitHub

1. Đăng nhập vào GitHub.
2. Nhấn vào dấu '+' ở góc trên bên phải và chọn **New repository**.
3. Điền các thông tin sau:
   - **Repository name:** `ktmt-nhapmon` (hoặc một tên khác tùy ý)
   - **Description (Optional):** Dự án đầu tiên của tôi.
   - Chọn **Public**.
   - Tích vào ô **Add a README file**. Việc này sẽ tạo sẵn một file văn bản giới thiệu về dự án.
4. Nhấn nút **Create repository**.

Bước 2: Sao chép (Clone) Repo về máy

1. Tại trang repo bạn vừa tạo trên GitHub, nhấn vào nút màu xanh <> **Code**.
2. Trong tab **HTTPS**, sao chép (copy) đường dẫn URL. Nó sẽ có dạng:
   `https://github.com/ten_cua_ban/ktmt-nhapmon.git`
3. Mở **Terminal** hoặc **Git Bash** trên máy tính của bạn.
4. Dùng lệnh `cd` để di chuyển đến thư mục bạn muốn lưu dự án (ví dụ: `cd Desktop`).
5. Chạy lệnh `git clone` và dán URL bạn vừa sao chép vào:
   ```
   git clone https://github.com/TEN_CUA_BAN/ktmt-nhapmon.git
   ```
6. Bây giờ, bạn sẽ thấy một thư mục mới tên là `ktmt-nhapmon` xuất hiện.

Bước 3: Thực hiện thay đổi đầu tiên

1. Dùng lệnh `cd` để di chuyển vào bên trong thư mục dự án:

   ```
   cd ktmt-nhapmon
   ```

2. Mở file `README.md` bằng một trình soạn thảo văn bản bất kỳ (như VS Code, Notepad++, Gedit...).
3. Thêm vào file một vài dòng giới thiệu về bản thân:
   - Họ và tên
   - Mã số sinh viên
   - Một câu giới thiệu ngắn
4. Lưu file lại sau khi chỉnh sửa.

Bước 4: Lưu lại thay đổi (Add & Commit)

Bây giờ, chúng ta sẽ báo cho Git biết về những thay đổi này.

1. Trong Terminal / Git Bash, chạy lệnh sau để kiểm tra trạng thái:
   `git status`
   Bạn sẽ thấy Git báo rằng file `README.md` đã bị thay đổi.
2. Dùng lệnh `git add` để chọn file `README.md` và đưa vào "khu vực chờ" (staging area):
   ```
   git add README.md
   ```
3. Dùng lệnh `git commit` để tạo một "bản ghi" cho thay đổi này, kèm theo một thông điệp mô tả rõ ràng:
   ```
   git commit -m "docs: Update README with personal information"
   ```
   Mẹo: Một thông điệp commit tốt thường bắt đầu bằng một động từ (Add, Update, Fix, Feat...) và mô tả ngắn gọn thay đổi.

Bước 5: Đẩy (Push) thay đổi lên GitHub

Bước cuối cùng là đồng bộ những commit mới của bạn lên server GitHub.

1. Trong Terminal / Git Bash, chạy lệnh sau:

   ```
   git push
   ```

   Lần đầu tiên, Git có thể yêu cầu bạn đăng nhập vào tài khoản GitHub.

2. **Kiểm tra kết quả:**
   - Quay trở lại trang repo của bạn trên trình duyệt web.
   - Tải lại (refresh) trang.
   - Bạn sẽ thấy nội dung của file `README.md` đã được cập nhật với thông tin bạn vừa thêm vào.

Chúc mừng! Bạn đã hoàn thành luồng làm việc cơ bản với Git và GitHub.

Tóm tắt và Định hướng

Những gì đã học:

- Hiểu được tầm quan trọng của việc quản lý phiên bản.
- Phân biệt được vai trò của Git và GitHub.
- Nắm được các khái niệm cốt lõi: Repository, Commit, Branch.
- Thực hành thành công luồng làm việc cơ bản: **Clone** -> **Edit** -> **Add** -> **Commit** -> **Push**.

Định hướng rèn luyện tiếp theo:

- **Thực hành thường xuyên:** Sử dụng Git/GitHub cho tất cả các bài tập, đồ án môn học sắp tới.
- **Tìm hiểu về Branching và Merging:** Đây là kỹ năng bắt buộc để làm việc nhóm hiệu quả.
- **Tìm hiểu về Pull Requests:** Quy trình chuẩn để đóng góp vào các dự án trên GitHub.
- **Khám phá các dự án mã nguồn mở:** Đọc code của người khác là một cách học tuyệt vời.

Bài tập nhóm - Giới thiệu

Mục tiêu:

- Rèn luyện kỹ năng sử dụng Git/GitHub trong môi trường nhóm.
- Kết hợp trách nhiệm cá nhân (mini-project) với phối hợp nhóm (pull request, review, merge).
- Làm quen với quy trình làm việc giống dự án phần mềm thực tế.

Nhiệm vụ cá nhân trong nhóm

Mỗi sinh viên cần:

- Tạo một nhánh riêng (`feature-[MSSV]`).
- Thêm một file/module cá nhân (code, tài liệu hoặc trang HTML).
- Thực hiện ít nhất 3 commit có ý nghĩa (ví dụ: `feat`, `fix`, `docs`).
- Ghi rõ thông tin cá nhân (Họ tên, MSSV) trong file của mình.

Nhiệm vụ nhóm

Cả nhóm cùng thực hiện:

- Tạo Pull Request từ nhánh cá nhân $\rightarrow$ `main`.
- Review chéo và comment trước khi merge.
- Thực hành giải quyết **conflict** (ít nhất một lần).
- Dùng `Issues` để phân chia công việc.
- Tạo tag phiên bản **v1.0** sau khi hợp nhất toàn bộ.

Nhiệm vụ nâng cao

- Thiết lập **GitHub Pages** để publish repo nhóm.
- Cấu hình cơ bản **GitHub Actions** (CI/CD).
- Thêm file `.gitignore` để quản lý file thừa.
- Sử dụng các lệnh nâng cao như `git log -oneline -graph`, `git revert`.

Báo cáo nộp: Link repo nhóm + ít nhất 3 PR đã merge + file `REPORT.md`.