Tổng Hợp Kiến Thức

1. Khởi Tạo Repository

- **Mục đích:** Biến thư mục hiện tại thành một Git repository mới (tạo thư mục `.git`).
- **Khi dùng:** Bắt đầu một dự án mới hoàn toàn trên máy của bạn.
- **Ví dụ:**

```
mkdir my-project
cd my-project
git init
```

2. Quy Trình Làm Việc Cơ Bản (Workflow)

Modify → Status → Add → Status → Commit → Status

- `git status`
  - **Mục đích: Lệnh quan trọng nhất!** Kiểm tra trạng thái hiện tại của các file trong Working Directory và Staging Area.
  - **Cho biết:**
    - `Untracked files:` File mới, Git chưa theo dõi.
    - `Changes not staged for commit:` File đã được Git theo dõi và bị sửa đổi, nhưng chưa đưa vào Staging Area.
    - `Changes to be committed:` File đã được đưa vào Staging Area, sẵn sàng để commit.
    - `nothing to commit, working tree clean:` Mọi thay đổi đã được commit.
  - **Ví dụ:**

```
git status
```

- `git add <file>` hoặc `git add .`
  - **Mục đích:** Đưa các thay đổi từ Working Directory vào Staging Area ("chuẩn bị" cho commit).
  - **Cách dùng:**
    - `git add <tên_file>`: Chỉ thêm file/thay đổi cụ thể.
    - `git add .`: Thêm TẤT CẢ thay đổi trong thư mục hiện tại và con. **(Cẩn thận khi dùng!)**
  - **Ví dụ:**

```
git add index.html
git add .
```

- `git commit -m "Thông điệp commit"`
  - **Mục đích:** Lưu "ảnh chụp" (snapshot) của Staging Area vào lịch sử Repository.
  - **Cách dùng:** Cung cấp thông điệp commit ngắn gọn, rõ ràng với cờ `m`.
  - **Ví dụ:**

```
git commit -m "Feat: Add basic user login form"
```

3. Commit Đúng Cách & Sửa Đổi Commit Cuối Cùng

- **Nguyên tắc Commit:**
  - **Atomic:** Mỗi commit giải quyết MỘT vấn đề/tính năng logic.
  - **Message rõ ràng:**
    - Tiêu đề (Subject) < 50 ký tự, thể mệnh lệnh (Ví dụ: `Fix: ...`, `Add: ...`, `Refactor: ...`).
    - (Nếu cần) Thân (Body) giải thích chi tiết hơn, cách tiêu đề 1 dòng trống.
- `git commit --amend`

- **Mục đích: Sửa đổi commit CUỐI CÙNG** (chỉ khi commit đó chưa được push lên remote!). Dùng để:
  - Sửa lỗi chính tả trong commit message.
  - Thêm các thay đổi nhỏ/file bị quên vào commit đó thay vì tạo commit mới.
- **Cách dùng:**
  - Sửa message: `git commit --amend -m "Message đúng"`
  - Thêm file/thay đổi (đã `git add`): `git commit --amend` (mở editor) hoặc `git commit --amend --no-edit` (giữ message cũ).
- ⚠️ **CẢNH BÁO: KHÔNG BAO GIỜ** amend commit đã được push và chia sẻ!
- **Ví dụ:**

```
# Vừa commit xong nhưng quên add file style.css
git add style.css
git commit --amend --no-edit

# Vừa commit xong nhưng gõ sai message
git commit --amend -m "Feat: Correctly add user login form"
```

4. Xem Lịch Sử Commit

- `git log`
  - **Mục đích:** Xem lịch sử các commit đã thực hiện.
  - **Hiển thị:** Hash, Author, Date, Message.
  - **Thoát:** Nhấn phím `q`.
- **Các tùy chọn phổ biến:**
  - `--oneline`: Hiển thị mỗi commit trên 1 dòng (hash rút gọn + message).
  - `n <số>` (ví dụ: `-n 5`): Giới hạn số lượng commit hiển thị.
  - `--graph`: Vẽ đồ thị (dạng text) biểu diễn các nhánh và merge (hữu ích khi học về branch).

- `--stat`: Hiển thị thống kê file nào thay đổi, bao nhiêu dòng thêm/xóa mỗi commit.
- `-p` hoặc `--patch`: Hiển thị chi tiết nội dung thay đổi (diff) của từng commit.

- **Ví dụ:**

```
git log
git log --oneline
git log -n 3 --stat
git log -p index.html # Xem lịch sử thay đổi chỉ của file index.html
```