**✅ Git ও GitHub কী?**

* **Git**: একটি ভার্সন কন্ট্রোল সিস্টেম (VCS)। এর মাধ্যমে তুমি একটি কোড প্রজেক্টে কী পরিবর্তন হয়েছে, কে করেছে, কখন করেছে—এইসব ট্র্যাক করতে পারো।
* **GitHub**: একটি অনলাইন প্ল্যাটফর্ম যেখানে তুমি Git দিয়ে করা প্রজেক্টগুলো সংরক্ষণ, শেয়ার, এবং সহযোগিতায় কাজ করতে পারো।

**১. ✅ Discard changes in file (পরিবর্তন বাতিল করা)**

**যখন কোনো ফাইলে পরিবর্তন করো এবং সেটা রাখতে না চাও, তখন discard করতে পারো:**

bash

CopyEdit

git checkout -- filename

👉 এটা ফাইলটিকে আগের অবস্থায় ফিরিয়ে নেয়।

**২. ✅ Create GitHub account**

১. <https://github.com> এ যাও  
২. **Sign up** বোতামে ক্লিক করো  
৩. ইমেইল, পাসওয়ার্ড দিয়ে একাউন্ট তৈরি করো  
৪. ইমেইল ভেরিফিকেশন করো

**৩. ✅ Clone repository locally (লোকাল কপি নেওয়া)**

GitHub এ থাকা কোনো প্রজেক্ট নিজের কম্পিউটারে আনতে:

bash

CopyEdit

git clone https://github.com/user/repository.git

**৪. ✅ Set remote repository (লোকাল প্রজেক্টের সাথে GitHub যুক্ত করা)**

যদি আগে থেকেই লোকাল প্রজেক্ট থাকে, তাহলে GitHub রেপোজিটরি যুক্ত করতে:

bash

CopyEdit

git remote add origin https://github.com/username/repo-name.git

👉 origin হচ্ছে রিমোটের নাম (ডিফল্ট)।

**৫. ✅ Push changes to GitHub (পরিবর্তন GitHub এ পাঠানো)**

bash

CopyEdit

git add .

git commit -m "Your commit message"

git push origin main

⚠️ "main" ব্রাঞ্চের নাম। কারো কারো ক্ষেত্রে "master" হতে পারে।

**৬. ✅ Fetch and merge changes from remote (রিমোট থেকে পরিবর্তন আনা)**

bash

CopyEdit

git fetch origin

git merge origin/main

👉 fetch শুধু রিমোটের পরিবর্তন আনে  
👉 merge করলে সেগুলো লোকালে যোগ হয়

সংক্ষিপ্তভাবে এক লাইনে:

bash

CopyEdit

git pull origin main

**৭. ✅ Create pull request (PR তৈরি করা)**

যখন তুমি পরিবর্তন করেছো এবং তা মূল রেপোজিটরিতে যুক্ত করতে চাও:

১. GitHub এ যাও  
২. তোমার ব্রাঞ্চ নির্বাচন করো  
3. "Compare & Pull Request" বোতামে ক্লিক করো  
৪. টাইটেল ও বর্ণনা দিয়ে PR সাবমিট করো

**৮. ✅ Create new branch (নতুন ব্রাঞ্চ তৈরি)**

bash

CopyEdit

git checkout -b new-feature

👉 এটি new-feature নামের একটি নতুন ব্রাঞ্চ তৈরি করে এবং সেখানেই চলে যায়।

**৯. ✅ Delete remote branch (রিমোট থেকে ব্রাঞ্চ ডিলিট করা)**

bash

CopyEdit

git push origin --delete branch-name

**🔟 ✅ Fork repository (অন্যের রেপো নিজের একাউন্টে কপি করা)**

১. GitHub এ অন্যের রেপোজিটরি খুলো  
২. উপরের ডান দিকে "Fork" বোতামে ক্লিক করো  
৩. এটি তোমার একাউন্টে একটি কপি তৈরি করবে

👉 তুমি এখন সেটি নিজে ক্লোন করে কাজ করতে পারো।

**১১. ✅ Fetch changes without merging (পরিবর্তন আনো, কিন্তু merge করো না)**

bash

CopyEdit

git fetch origin

👉 এতে শুধু নতুন পরিবর্তন ডাউনলোড হবে, কিন্তু লোকালে মিশবে না।

**১২. ✅ Clone with latest commits only (শুধু শেষ কিছু কমিট ক্লোন করা)**

bash

CopyEdit

git clone --depth 1 https://github.com/user/repo.git

👉 এটা shallow clone করে, অর্থাৎ পুরনো ইতিহাস আনবে না, শুধু সর্বশেষ অবস্থা।

**১৩. ✅ View file modification history (ফাইলে কে কখন কী পরিবর্তন করেছে দেখো)**

bash

CopyEdit

git log filename

👉 শুধু ওই ফাইলের পরিবর্তনের ইতিহাস দেখাবে।

**🔁 Bonus: Git Workflow Recap (ধারাবাহিক কাজের ধরন)**

১. Git init / Git clone  
২. কোড পরিবর্তন  
৩. git add .  
৪. git commit -m "message"  
৫. git push origin branch-name  
৬. PR তৈরি  
৭. কোড review ও merge  
৮. নতুন ব্রাঞ্চে কাজ চালিয়ে যাও

**✅ যদি এক লাইনে বলি:**

| **কাজ** | **কমান্ড** |
| --- | --- |
| নতুন প্রজেক্টে Git শুরু | git init |
| GitHub থেকে ক্লোন | git clone URL |
| ফাইল Stage করা | git add filename বা git add . |
| কমিট করা | git commit -m "message" |
| রিমোট যুক্ত করা | git remote add origin URL |
| GitHub এ পাঠানো | git push origin main |
| নতুন ব্রাঞ্চ তৈরি | git checkout -b branch-name |
| ব্রাঞ্চ ডিলিট (রিমোটে) | git push origin --delete branch-name |
| ফাইল discard করা | git checkout -- filename |
| ফাইল ইতিহাস দেখা | git log filename |

**What does the command git status do?**

The git status command is used to:

* Show which **branch** you're on
* List **staged**, **unstaged**, and **untracked** files
* Indicate if there are any changes that need to be committed

It does **not**:

* Only show the branch (1 is incomplete)
* Commit changes (that's git commit, so 3 is wrong)
* Delete anything (4 is definitely wrong)

**What is GitHub?**

### Explanation:

* **GitHub** is a **web-based platform** that uses **Git**, a version control system, to help developers **host, manage, and collaborate on code**.
* It provides tools for **issue tracking, pull requests, code review, CI/CD**, and more.
* While **Git** itself is the version control system (so 1 is not entirely wrong conceptually), GitHub is **not** Git — it's a **service built on top of Git**.
* It is **not** a file storage service like Dropbox or Google Drive (so 2 is wrong), and it's definitely **not** a programming language (4 is wrong).