
Lecture: Git, GitHub

১. Git কী এবং কেন ব্যবহার করব?

? Git কী?

Git হলো এক ধরনের **version control system** — মানে আপনি যদি একটি প্রজেক্টে কাজ করেন, তাহলে Git সেই প্রজেক্টের প্রতিটি পরিবর্তনের হিসাব রাখে।

উদাহরণ:

ভাবুন, আপনি একটি গল্লের বই লিখছেন। প্রতিদিন কিছু না কিছু লিখেন। আপনি চাইবেন পুরনো লেখা গুলো হারিয়ে না যাক। তাই আপনি প্রতিদিন একটা কপি করে রাখেন। Git এই কাজটাই অটোমেটিক করে দেয়।

✓ কেন Git ব্যবহার করবেন?

- পরিবর্তনের ইতিহাস পাওয়া যায়
- দলগতভাবে কাজ করা সহজ হয়
- পুরনো ভার্সনে ফিরে যাওয়া যায়
- নতুন ফিচার আলাদাভাবে পরীক্ষা করা যায় (branch)

Git = আপনার প্রজেক্টের টাইম ট্রান্ডেল মেশিন!

📁 ২. ভার্সন কন্ট্রোল কী?

সহজ ভাষায়:

Version control system হলো এমন একটি সফটওয়্যার যা প্রতিটি ফাইলে কে, কখন, কী পরিবর্তন করেছে — এই হিসাব রাখে।

উদাহরণ:

ভাবুন আপনি আপনার ঘর সাজাচ্ছেন। আপনি প্রতিদিন নতুন কিছু যোগ করছেন। হঠাৎ

একদিন মনে হলো আগের সাজানোটা ভালো ছিল। যদি ছবি তুলে রাখতেন প্রতিদিন, তাহলে আগের অবস্থায় ফিরে যাওয়া যেত। Git ঠিক এই রকম ছবি তোলে প্রতিদিনের কাজের।

🛠️ ৩. সাধারণ Git কমান্ডসমূহ

কমান্ড	কাজ
git init	প্রজেক্টে git চালু করা
git add file.py	নির্দিষ্ট ফাইল স্টেজে তোলা
git commit -m "msg"	চেঞ্চগুলো সংরক্ষণ করা
git status	কী কী পরিবর্তন হয়েছে দেখা
git log	পুরনো commit এর তালিকা দেখা

উদাহরণ:

ধরি আপনি একজন দোকানদার। প্রতিদিন আপনি কী কিনলেন, বিক্রি করলেন সেটা একটা খাতায় লেখেন। `git commit` ঠিক সেই হিসাব লেখা।

৪. GitHub কী এবং কিভাবে কাজ করে?

? GitHub কী?

GitHub হলো Git প্রজেক্টের ক্লাউড হোস্টিং ওয়েবসাইট। এটা Git-এ করা কাজগুলো ইন্টারনেটে রাখে যেন আপনি এবং আপনার টিম মেম্বাররা একসাথে কাজ করতে পারেন।

উদাহরণ:

আপনি আপনার মোবাইলে ছবি তোলেন (Git), আর Google Photos-এ রেখে দেন (GitHub)। তাহলে অন্য মোবাইল থেকেও দেখতে পারবেন।

📁 ৫. GitHub-এ রেপোজিটরি তৈরি ও ব্যবস্থাপনা

ধাপসমূহ:

- [github.com](#) ঘান
- “New Repository” চাপুন

3. নাম দিন, README চাই কিনা ঠিক করুন
4. এরপর Git কমান্ড দিয়ে যুক্ত করুন:

```
git remote add origin https://github.com/username/repo.git  
git push -u origin main
```

রিপোজিটরি = প্রজেক্টের প্যাকেট

এ. কোড ক্লোন ও পুশ করা

ক্লোন করা:

```
git clone https://github.com/username/repo.git
```

উদাহরণ:

কোনো দোকান থেকে একটা ডিজাইন কিনে আপনি নিজের দোকানে আনলেন।

কোড পুশ করা:

```
git add .  
git commit -m "my update"  
git push origin main
```

উদাহরণ:

আপনার দোকানের ডিজাইন আবার কারখানায় পাঠালেন বানাতে।

৭. Branch ও Merge এর সহজ ধারণা

Branching:

- মানে নতুন রাস্তায় কাজ করা — মূল রাস্তা (main) ঠিক রেখে নতুন ফিচার এক্সপেরিমেন্ট করা

কমান্ড:

```
git branch feature-1  
git checkout feature-1
```

Merging:

```
git checkout main  
git merge feature-1
```

উদাহরণ:

ধরি আপনি নতুন এক রাস্তা ট্রাই করলেন (branch), সেটা ভালো হলে সেটা রেস্টুরেন্টের মেনুতে যুক্ত করলেন (merge)!

ই ৮. একটি আদর্শ প্রজেক্ট ফোল্ডার স্ট্রাকচার

```
myproject/  
|  
|   src/          # মূল কোড  
|   tests/        # টেস্ট ফাইল  
|   docs/         # ডকুমেন্টেশন  
|   data/         # ডেটা  
|   scripts/      # হেল্পার স্ক্রিপ্ট  
|   README.md     # প্রজেক্ট সম্পর্কে তথ্য  
|   .gitignore    # কোন ফাইলগুলো Git বাদ দেবে  
|   requirements.txt # কোন লাইব্রেরি দরকার
```

উদাহরণ:

এটা অনেকটা স্কুলের ক্লাস রুটিনের মতো — প্রতিটি বিষয়/ফোল্ডার আলাদা এবং পরিষ্কার।

৯. কোড বেস সংগঠনের সেরা প্র্যাকটিস

✓ করণীয়:

- ফোল্ডারগুলো পরিষ্কার রাখুন
- ছোট ছোট কমিট করুন, অর্থপূর্ণ মেসেজ দিন
- .gitignore ব্যবহার করুন
- README.md লিখে প্রজেক্ট ব্যাখ্যা দিন
- main বাঞ্চে সরাসরি কাজ না করে branch ব্যবহার করুন

✗ যা করবেন না:

- বড় ডেটা ফাইল Git-এ রাখবেন না
- কোনো ফাইল ভুলে commit করে ফেলবেন না
- বারবার force push করবেন না



সারাংশ

বিষয়	সারাংশ
Git	লোকাল ভার্সন কন্ট্রোল টুল
GitHub	Git এর ফ্লাউড হোস্ট
Git কমান্ড	init, add, commit, push, branch, merge
Version Control	কাজের ইতিহাস রাখা এবং দলগত কাজ সহজ করা
ফোল্ডার স্ট্রাকচার পরিষ্কার ও মডুলার রাখা উচিত	
Best Practices	ব্রাঞ্চিং, কমিট বার্তা, ডকুমেন্টেশন
