# git and github

### ১. Git (গিট) - এটি একটি সফটওয়্যার/টুল

Git হলো একটি **Version Control System (VCS)**। ধরুন, আপনি একটি প্রোজেক্টে কাজ করছেন এবং প্রতিদিন নতুন নতুন কোড যোগ করছেন।

- **কেন ব্যবহার করবেন?** যদি আজ আপনার কোডে কোনো ভুল হয় এবং আপনি চান গত পরশুদিন কোড যেমন ছিল সেখানে ফিরে যেতে, তবে Git আপনাকে সেই সুযোগ দেবে।
- **কোথায় থাকে?** এটি আপনার নিজের কম্পিউটারে ইনস্টল করা থাকে এবং ইন্টারনেটের প্রয়োজন হয় না।
- **মূল কাজ:** এটি আপনার ফাইলের সব পরিবর্তন (change) ট্রাক করে।

### ২. GitHub (গিটহাব) - এটি একটি অনলাইন প্ল্যাটফর্ম

GitHub হলো একটি ওয়েবসাইট যেখানে আপনি আপনার Git দিয়ে করা কাজগুলো জমা রাখতে পারেন।

- **মূল কাজ:** এটি আপনার কোডের জন্য একটি ক্লাউড স্টোরেজ বা হোস্টিং সার্ভিস।
- **কেন ব্যবহার করবেন?** আপনার কম্পিউটার নষ্ট হয়ে গেলেও আপনার কোড এখানে নিরাপদ থাকবে। এছাড়া এটি বিশ্বের সবচেয়ে বড় ডেভেলপার কমিউনিটি, যেখানে অন্যরা আপনার কোড দেখতে পারে বা আপনি অন্যের প্রজেক্টে সাহায্য করতে পারেন।
- **মালিকানা:** এটি বর্তমানে **Microsoft**-এর মালিকানাধীন।

### ৩. GitLab (গিটল্যাব) - এটি GitHub-এর মতোই আরেকটি প্ল্যাটফর্ম

GitLab এবং GitHub প্রাই একই কাজ করে। তবে এদের মধ্যে কিছু ছোটখাটো পার্থক্য আছে:

- **মূল পার্থক্য:** GitLab সাধারণত বড় বড় কোম্পানি বা প্রফেশনাল কাজের জন্য বেশি জনপ্রিয় কারণ এটিতে **CI/CD (Continuous Integration and Deployment)** বা কোড অটোমেশন ফিচারগুলো আগে থেকেই খুব শক্তিশালীভাবে দেয়া থাকে।
- **প্রাইভেসি:** অনেক কোম্পানি তাদের নিজেদের সার্ভারে GitLab সেটআপ করে ব্যবহার করতে পছন্দ করে।

**একনজরে তুলনা:**

| **বিষয়** | **Git (গিট)** | **GitHub / GitLab** |
|---|---|---|
| **ধরন** | এটি একটি সফটওয়্যার। | এগুলো ক্লাউড সার্ভিস/ওয়েবসাইট। |
| **কাজ** | কোডের ভার্সন কন্ট্রোল করা। | কোড অনলাইনে জমা রাখা এবং শেয়ার করা। |
| **ইন্টারনেট** | প্রয়োজন নেই। | প্রয়োজন আছে। |
| **ইনস্টলেশন** | পিসিতে ইনস্টল করতে হয়। | অনলাইনে অ্যাকাউন্ট খুলতে হয়। |

সহজ উদাহরণ:

মনে করুন, আপনি একটি বই লিখছেন।

- **Git** হলো আপনার কম্পিউটারের সেই ফাংশন যা আপনার বইয়ের প্রতিটি সেভ করা ভার্সন (যেমন: ১ম ড্রাফট, ২য় ড্রাফট) মনে রাখে।

- **GitHub বা GitLab** হলো আপনার সেই অনলাইন ড্রাইভ বা ডায়েরি যেখানে আপনি আপনার বইয়ের ড্রাফটগুলো আপলোড করে রাখছেন যাতে আপনার বন্ধুরাও সেটা পড়তে পারে এবং এডিট করতে সাহায্য করতে পারে।

সহজ কথায় বলতে গেলে, **Git** হলো আপনার কাজের হিসাব রাখার একটি ডায়েরি, আর **GitHub** বা **GitLab** হলো সেই ডায়েরিটি অনলাইনে রাখার আলমারি।

### 🔹 Git কী?

**Git = Version Control System (VCS)**

👉 সহজ কথায়

কোডের **ইতিহাস (history)** রাখে
কে কখন কী পরিবর্তন করলো ট্র্যাক করে
টিমে কাজ করা সহজ করে

📌 ধরো তুমি প্রতিদিন কোড লিখছো

- আজ ভাঙলো ❌
- গতকালের ভালো কোডে ফিরে যেতে চাও ✅
  ➡ Git এখানেই কাজ করে

### 🔹 GitHub কী?

**GitHub = Cloud platform যেখানে Git repository থাকে**

| Git | GitHub |
|---|---|
| Local tool | Online service |

| Computer-এ চলে | Internet-এ |
| --- | --- |
| Version control | Code hosting + collaboration |

👉 Git ছাড়া GitHub নেই

👉 GitHub ছাড়া Git চলতে পারে

## 🔹 Git কেন দরকার?

✓ Code history

✓ Undo / rollback

✓ Team collaboration

✓ Branching

✓ Open source contribution

## 🔹 Git Architecture (Mind Map)

Working Directory

↓ git add

Staging Area

↓ git commit

Local Repository

↓ git push

Remote Repository (GitHub)

## 🔹 Git Install

git --version

### 🔹 Git Configuration (প্রথম কাজ)

git config --global user.name "Arman Rakib"

git config --global user.email "arman@email.com"

### 🔹 চেক করো:

git config --list

### 🔹 Repository কী?

👉 **Project folder যেটা Git track করে**

## 🟢 নতুন Project শুরু করা

git init

📁 .git folder তৈরি হবে (এটাই Git brain 🧠)

## 🟢 Git Status (সবচেয়ে বেশি ব্যবহার হবে)

git status

দেখায়:

- কোন file untracked
- কোন staged
- কোন modified

## 🟢 File Add (Staging)

git add file.txt  # single file

git add .  # সব file

## 🟢 Commit (Snapshot)

git commit -m "Initial commit"

👉 commit = একটা **checkpoint**

## 🔹 Git Lifecycle

| State | Meaning |
|---|---|
| Untracked | Git জানে না |
| Modified | Change হয়েছে |
| Staged | Commit করার জন্য ready |
| Committed | Safe snapshot |

## 🔹 Commit History দেখার Command

git log

git log --oneline

git log --graph

## 🔹 File Undo / Reset

### 🟡 Modified file reset

git checkout -- file.txt

### 🟡 Staged file unstaged

git reset file.txt

### 🟡 Last commit undo

git reset --soft HEAD~1

git reset --hard HEAD~1 ❌ danger

### 🔹 Branch কী?

👉 আলাদা timeline
👉 Main code নষ্ট না করে experiment

### 🟢 Branch Command

git branch # সব branch

git branch dev # create

git checkout dev # switch

git checkout -b feature # create + switch

### 🟢 Merge Branch

git checkout main

git merge dev

### 🔴 Merge Conflict

যখন Git বুঝতে পারে না কোনটা রাখবে

👉 manually fix
👉 তারপর:

git add .

git commit

🔹 **GitHub ব্যবহার (Step by Step)**

**1. GitHub repo বানাও**

- github.com
- New Repository

**2. Local → GitHub connect**

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

চেক:

git remote -v

**3. Push**

git push -u origin main

**4. Next time push**

git push

🔹 **Clone (Existing Project)**

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

- **Pull vs Fetch**

| Command | কাজ |
| --- | --- |
| git fetch | শুধু update আনে |
| git pull | fetch + merge |

git pull origin main

- **Git Ignore**

যে file Git track করবে না

.gitignore

node_modules/

.env

dist/

- **Tag (Release version)**

git tag v1.0

git push origin v1.0

- **Stash (Temporary save)**

git stash

git stash pop

- **Git Rebase (Advanced)**

git rebase main

✓ Clean history
❌ risky if shared branch

### 🔹 Git Cherry-pick

git cherry-pick commitID

### 🔹 GitHub Collaboration Flow

Fork → Clone → Branch → Commit → Push → Pull Request

### 🔹 Pull Request (PR)

👉 Code review
👉 Team approval
👉 Merge to main

### 🔹 Interview Must-Know Q&A

**Q: git vs github?**
👉 Tool vs platform

**Q: git pull vs git fetch?**
👉 merge আছে / নাই

**Q: merge vs rebase?**
👉 safe vs clean history

- **Daily Git Workflow (Real Life)**

```
git pull

git checkout -b feature

# code

git add .

git commit -m "feature added"

git push origin feature

# PR
```