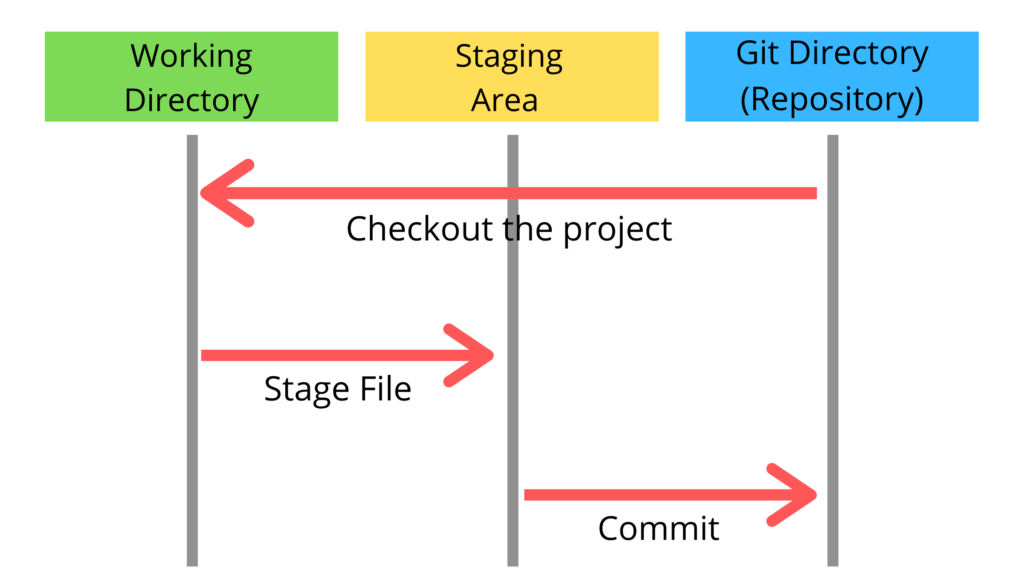
**GIT TUTORIAL**



1. **git config --global user.name "Your Name"**

**git config --global user.email** [**your\_email@example.com**](mailto:your_email@example.com)

* Sets your Git identity. Use once on your machine so all commits show your name & email.

1. **git init** – Initialize repo

* Creates a new Git repository in the current folder.(right click in folder select git bash here) Use this if starting a fresh project.

1. **git clone [Project SSH key Copy and paste it here]**

**git clone [Project SSH key Copy and paste it here] .**

(. 🡪 it clone whole repo with folder name we don’t want to create a folder)

* Copies an existing repo from GitHub to your machine. Use at the start of working on an existing project.

1. **git status**

* Shows which files are modified, staged, or untracked. Use very often before/after making changes.

1. **git add .** # adds all changes

**git add filename**

* Moves files into the staging area (ready for commit). Use after editing/creating files.

1. **git commit -m "Your commit message"**

* Saves a snapshot of staged changes in history. Use after staging with git add.

1. **git log**

* Shows past commits. Use when you want to check project history.

**git log --oneline –graph**

* --oneline → compresses each commit to one line (short hash + commit message).
* --graph → adds a text-based graph showing branch and merge history.

**Why we use origin?**

* origin is just a **nickname (alias)** for the remote repository URL on GitHub.
* It’s used so we don’t have to type the full GitHub URL every time.
* Example: Apne code ko **GitHub me push karne ke liye** hume local repo ko GitHub repo se link karna padta hai → ye kaam origin karta hai.

**Steps to Connect GitHub Repo (Set Origin)**

**Step 1: Create a New Repository on GitHub**

* Go to GitHub → Click on **New Repository**.
* Example repo name: git-Tutorial.
* Keep it **empty** (no README, no .gitignore) so it won’t conflict with your local repo.

**Step 2: Setup SSH Key (One-time only)**

To securely connect your PC with GitHub without entering username/password:

1. **Generate SSH key**

**ssh-keygen -t rsa -b 4096 -C "your\_email@example.com"**

→ Just keep pressing **Enter** until it finishes.  
→ It creates two files inside ~/.ssh/:

* + id\_rsa → private key (never share this!)
  + id\_rsa.pub → public key (this one goes to GitHub)

1. **Start SSH agent**

**eval "$(ssh-agent -s)"**

1. **Add the private key to agent**

**ssh-add ~/.ssh/id\_rsa**

1. **Copy the public key**

**cat ~/.ssh/id\_rsa.pub**

Copy the whole line (starts with ssh-rsa …).

1. **Add SSH key to GitHub**
   * Go to **GitHub → Profile → Settings → SSH and GPG Keys → New SSH key**.
   * Paste your public key and save.

**Step 3: Link Local Repo with GitHub (Set Origin)**

Now we tell Git that our GitHub repo is the “origin”:

**git remote add origin** [**git@github.com:your-username/git-Tutorial.git**](mailto:git@github.com:your-username/git-Tutorial.git)

**Step 4: Push Your Code**

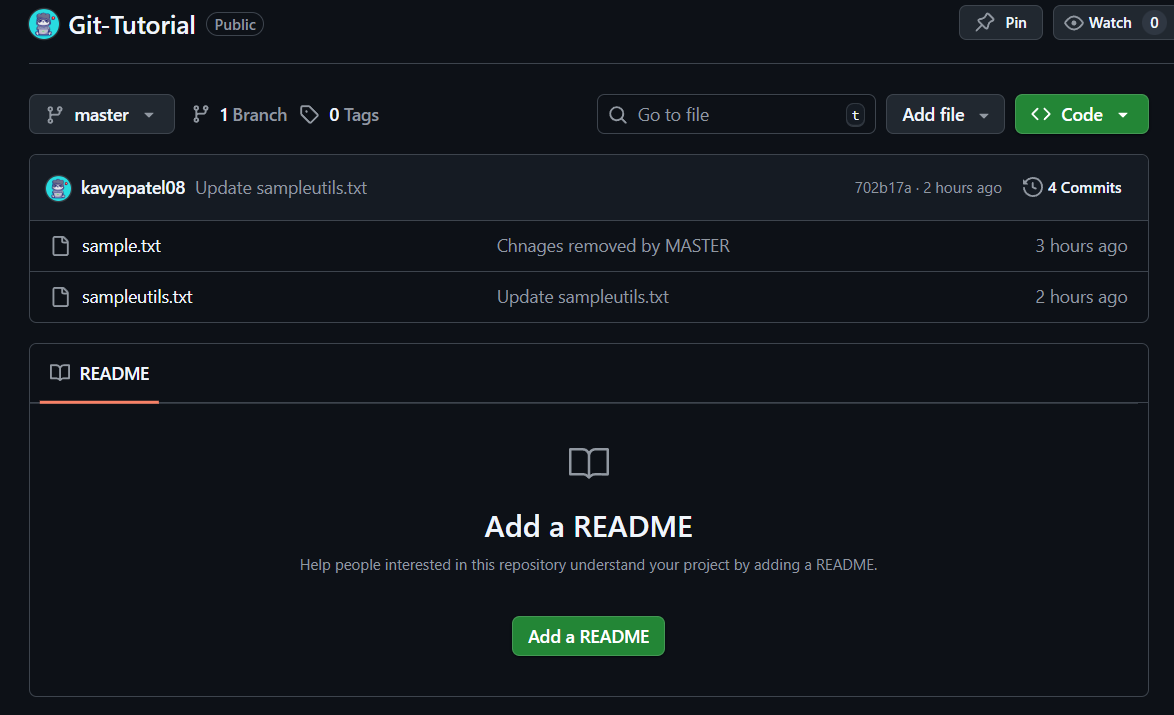
Finally, push your local commits to GitHub:

git branch -M main # rename branch to main (if not already)

git push -u origin main # push local main → GitHub main

I use in my base –> **git push -u origin master**

Output:

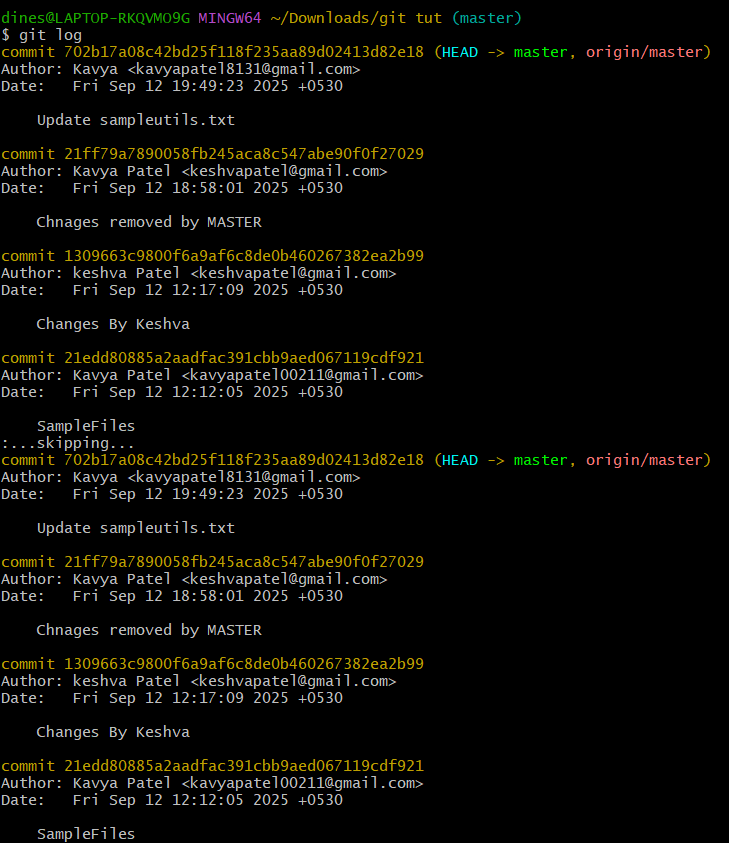


Now change any one of the file and write in commit message

Now go to your bash

**Git push -u origin master**

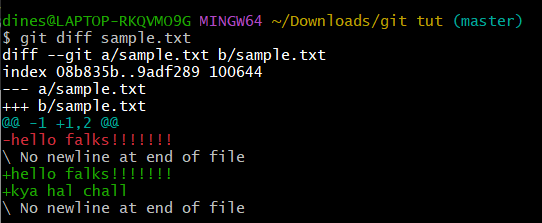
**Git log**



**git diff filename**

✔️ Shows the line-by-line changes you made in a specific file compared to the last commit.

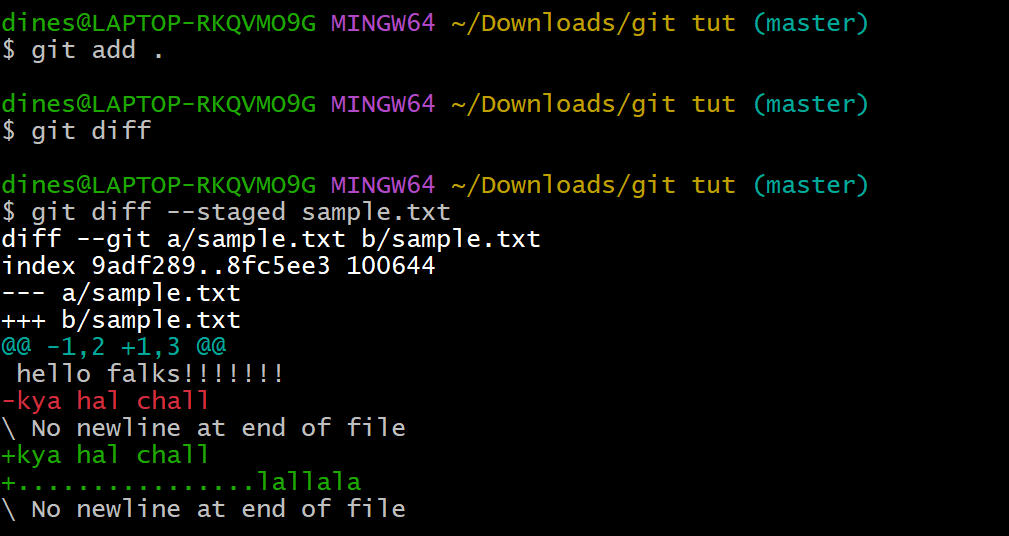
✔️ Use it before staging/committing to review what exactly changed in that file.



**git diff --staged sample.txt**

✔️ Shows the changes in sample.txt that are already staged (added with git add) compared to the last commit.

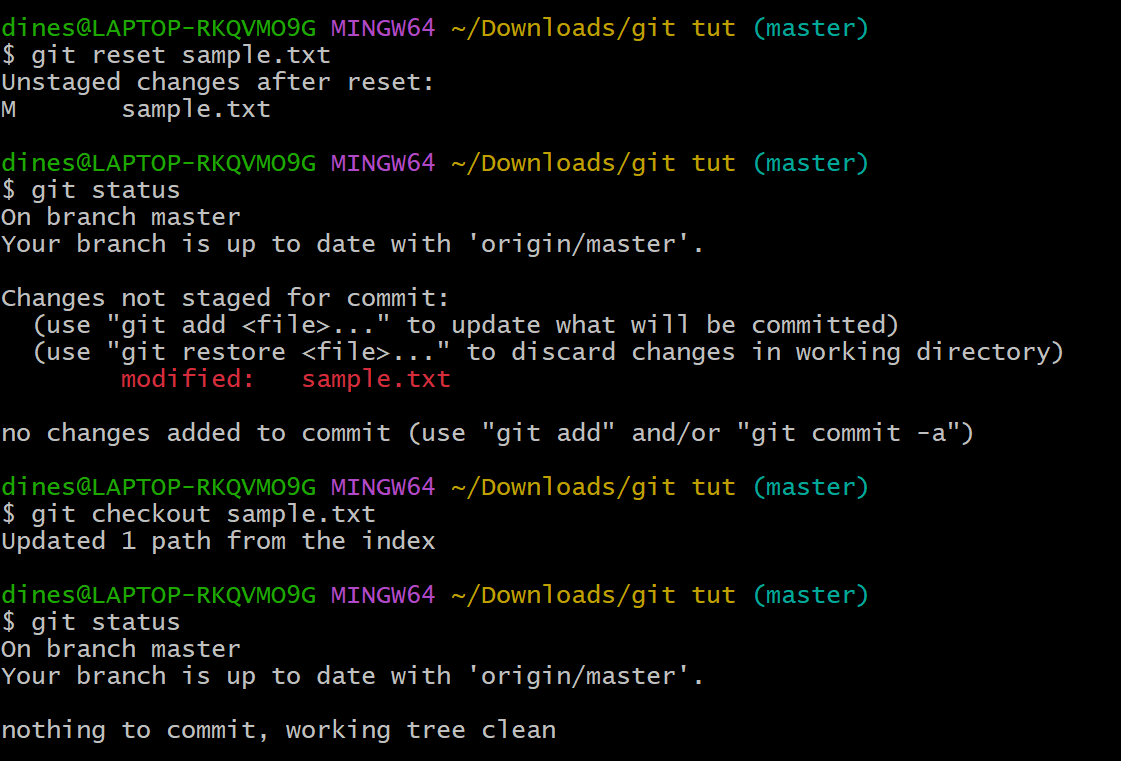
✔️ Use it to review staged changes before committing.



* **git reset sample.txt** → Unstages the file (removes it from staging area) but keeps the changes in the working directory.
* **git checkout sample.txt** → Discards local changes in the file and restores it to the last committed version.
* **git checkout .** → discards **all local changes** in the current directory(All files) and restores files to the last committed state.

👉 In short:

* reset → *unstage, keep changes*
* checkout → *remove changes, restore file*



| **Command** | **What it does (simple words)** |
| --- | --- |
| **git add -A** | Stages **all changes** → new, modified, and deleted files (from the whole repo). |
| **git add .** | Stages **new + modified files** in the current folder, but won’t stage deletions outside it. |
| **git add -u** | Stages **modified + deleted files only**, no new untracked files. |

We use **.gitignore** to tell Git which files/folders to ignore (not track).  
This helps keep the repo clean by skipping temp files, secrets, and system-generated files.

* 1. Create empty file:

**touch .gitignore**

* 1. Open in VS Code:

**code .gitignore**

* 1. Open in Notepad:

**notepad .gitignore**

After creating, you can add patterns like:

\*.log

\*.tmp

node\_modules/

\_\_pycache\_\_/

.env

* 1. **git rm --cached FILENAME**

removes the file from Git’s tracking (staging/repo) but keeps it safe in your local folder

(aap kisi file ko bohot der tak track kar rahe the or bad me apko pata chala ki use track karne ka koi matlab nahi hai aese me agar us file ko hum .gitignore me dalenge to uski history already rahegi and vo remove karne ke liye hume ye cmd run karna padega)

for more info 🡪 <https://git-scm.com/book/en/v2/Git-Basics-Recording-Changes-to-the-Repository#_ignoring>

**MIMP - Git Branch**

**1. git branch login-system**

👉 Creates a new branch called login-system.  
(It’s like making a copy of your project where you can try new features without disturbing master.)

**2. git checkout login-system**

👉 Switches your working directory to the new login-system branch.  
(You are now working inside this branch.)

**3. git push -u origin login-system**

👉 Pushes your new branch to GitHub.  
The -u sets tracking, so next time you can just do git push without writing branch name.

**4. git merge login-system**

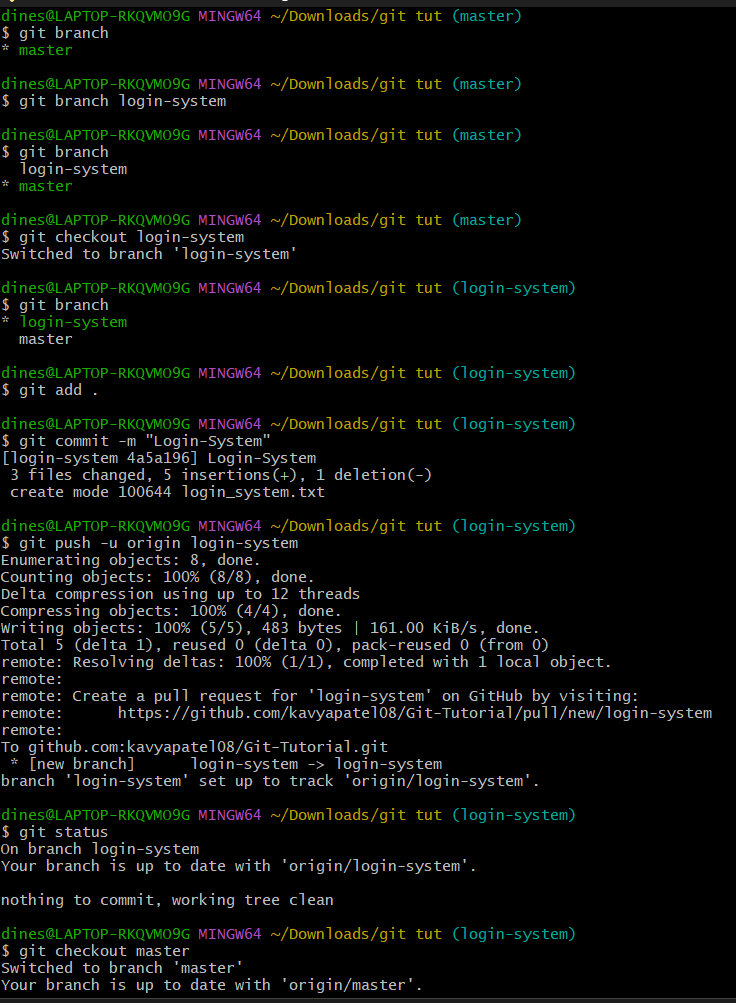
👉 Combines changes from login-system into the master branch.  
Here it was a fast-forward merge, meaning master just moved forward to include new commits.

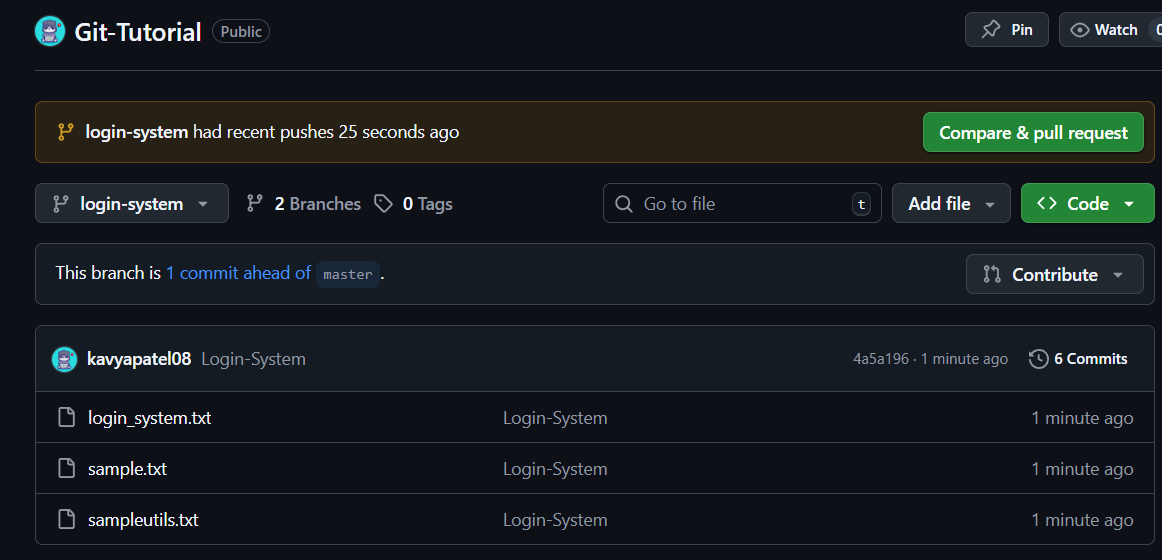
**5. git branch -d login-system**

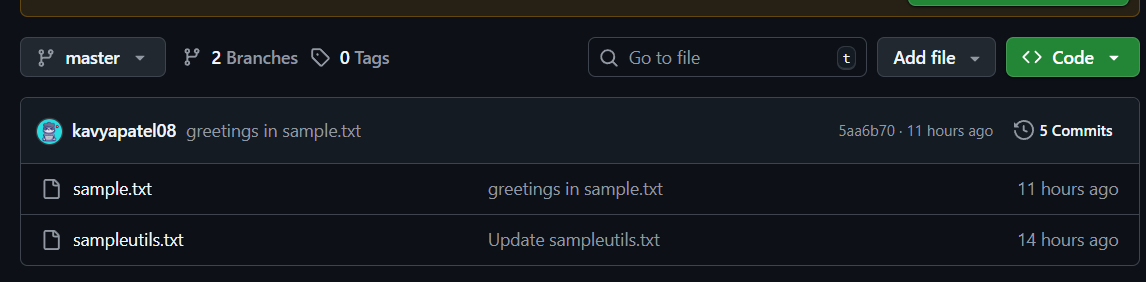
👉 Deletes the branch **locally** after merging.  
Since changes are already in master, you don’t need it anymore.

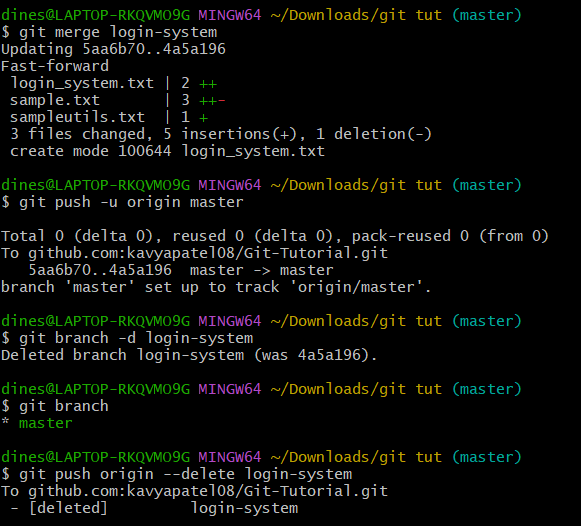
**6. git push origin --delete login-system**

👉 Deletes the branch **from GitHub (remote)**.  
This keeps your repo clean and avoids unused branches piling up.









You can see only 1 branch after merging it

