Git & GitHub — Comprehensive Guide for Beginners

Purpose: This note explains what Git and GitHub are, why we use them, and shows practical, copy-paste commands and workflows your students can use to work on the class repository and collaborate safely.

1. Quick overview — Git vs GitHub

- **Git** is a distributed version control system that runs on your computer. It tracks changes to files, lets you keep history, branch, merge, and revert changes.
- **GitHub** is a cloud service that hosts Git repositories, provides a web UI for collaboration (Pull Requests, Issues), and adds features like Actions, Pages, and code review.

Analogy: Git is the engine in your car; GitHub is the garage where you park, show & share your car with others.

2. Why use version control?

- Keep a history of all changes (who changed what and when).
- Work on features in isolation using branches.
- Collaborate safely (pull requests + code review).
- Revert mistakes without panic.

3. Setup (one-time per computer)

Install Git

- **Windows:** Download and install from `https://git-scm.com/download/win` (use Git Bash).

- **macOS:** `brew install git` (if Homebrew installed) or download from `https://git-scm.com`.

- **Linux (Debian/Ubuntu):** `sudo apt update && sudo apt install git`

- Confirm: `git --version`

Configure Git (replace with your info)

```
git config --global user.name "Firstname Lastname"
git config --global user.email "youremail@example.com"
```

```
# optional: set default editor
git config --global core.editor "code --wait"
```

Create (or sign in to) a GitHub account

- Go to `https://github.com/` and sign up using a username you'll use for class. Keep it simple and professional.

Authentication methods (recommended)

1. **SSH (recommended for CLI):** more convenient and secure after setup.
   - Generate key: `ssh-keygen -t ed25519 -C "youremail@example.com"` (or `rsa` if needed)
   - Start ssh-agent and add key: `eval "$(ssh-agent -s)"` then `ssh-add ~/.ssh/id_ed25519`
   - Copy public key and add to GitHub → Settings → SSH and GPG keys.
   - Test: `ssh -T git@github.com`
2. **HTTPS with Personal Access Token (PAT):** GitHub stopped accepting account passwords. You can create a PAT and use it as your password when prompted.

4. Basic local workflow (commands you will use every day)

```
# clone the repo (only once per machine)
git clone [REPO_URL]
cd <repo-folder>

# check status
git status

# create and switch to a new branch
git checkout -b firstname-lastname-githubusername

# add new/changed files to the staging area
git add path/to/file.py
# or add everything:
git add .

# commit changes with a message
git commit -m "Add assignments - Firstname Lastname"

# push branch to remote for the first time
git push -u origin firstname-lastname-githubusername

# later: fetch remote changes and update local main
git checkout main
git pull origin main
```

```
# bring main into your branch
git checkout firstname-lastname-githubusername
git merge origin/main
# or
# git rebase origin/main    (advanced — avoid rebasing public branches unless
instructed)
```

Note: Always create a branch for your work — do not commit directly to `main`.

5. Branching & merging explained

- **Branches** let you work on features independently: `git branch` lists branches.
- Create branch: `git checkout -b feature-name`.
- Switch branch: `git checkout branch-name`.
- **Merging**: `git checkout main` then `git merge feature-branch` to bring changes into main.
- **Fast-forward vs. merge commit:** If main hasn't changed since branch was created, merge can be fast-forward. Otherwise a merge commit is created.
- **Conflicts:** If both branches changed the same line, Git will stop and ask you to resolve conflicts. Fix files, then `git add` and `git commit` to finish the merge.

6. Pull Request (PR) workflow — how we'll collaborate on GitHub

1. Create a branch locally: `git checkout -b firstname-lastname-githubusername`.
2. Add files inside the root folder named exactly as your branch (this is our class rule).
3. Commit & push: `git push -u origin firstname-lastname-githubusername`.
4. On GitHub, open a PR from your branch → base `main`. Use the required title and description template.
5. CI / Actions may run (in this class we use an auto-merge action). If the PR passes checks and meets rules, it will auto-merge.

PR title template: `Add assignments - Firstname Lastname`
PR description template:

```
- Name: Firstname Lastname
- GitHub: @your-github-username
- Files added:
  - firstname-lastname-githubusername/assignment1.py
  - firstname-lastname-githubusername/assignment2.py
- Notes (optional):
```

7. Undoing mistakes — safe ways to backtrack

- Undo changes in working directory (uncommitted): `git restore filename` or `git checkout -- filename` (older Git).
- Unstage a file: `git reset filename`.
- Amend last commit (before pushing): `git commit --amend` (be careful — don't amend public commits).
- Revert a commit already in history: `git revert <commit-hash>` — this creates a new commit that undoes changes.
- Reset (dangerous if not understood): `git reset --soft <commit>` / `--mixed` / `--hard` — do not use `--hard` on shared branches unless you know what you're doing.
- Stash uncommitted work: `git stash` and retrieve later with `git stash pop`.

8. Common errors & how to fix them

1) Authentication failed when pushing

- If using HTTPS: use a **Personal Access Token (PAT)** as your password.

- If using SSH: ensure your SSH key is added to GitHub and `ssh-agent` is running.

2) `git push` rejected (protected branch)

- You cannot push to `main` — create a branch and open a PR instead.

3) Merge conflicts

- Git will mark conflicted files with `<<<<<<<`, `=======`, `>>>>>>>` sections. Open the file, decide which lines to keep, remove markers, then `git add` and `git commit`.

4) `detached HEAD` state

- This means you are on a commit rather than a branch. Create a branch from this state: `git checkout -b new-branch`.

5) Large files causing push errors

- GitHub has file size limits. Use Git LFS or avoid pushing huge binaries.

9. Best practices (teacher's rules for the class)

- **Branch naming:** `firstname-lastname-githubusername` (all lowercase, hyphens).

- **Folder naming:** same as branch name, at repo root.

- **Commit messages:** short imperative verb (`Add`, `Fix`, `Update`), then context. Example: `Add assignments - John Doe` or `Fix assignment2 runtime error`.
- **One feature per branch / PR.** Keep PRs small and focused.
- **Don't change other folders** — only add files inside your personal folder.
- **Run your code locally** before pushing. If your assignments include scripts, make sure they run.

10. Useful `.gitignore` for Python projects

Create a `.gitignore` file in the repo root (if not present). Example entries:

```
# Byte-compiled / cache
__pycache__/
*.py[cod]

# Virtual environments
venv/
.env

# VSCode
.vscode/

# Mac
.DS_Store
```

11. Git & GitHub cheat sheet (quick commands)

Action	Command
Check Git version	`git --version`
Clone repo	`git clone [REPO_URL]`
New branch	`git checkout -b branch-name`
Switch branch	`git checkout branch-name`
Stage files	`git add path/to/file` or `git add .`
Commit	`git commit -m "message"`
Push branch	`git push -u origin branch-name`
Pull latest main	`git checkout main` then `git pull origin main`
Fetch remote PRs	`git fetch origin`
View log	`git log --oneline --graph --all`

Action	Command
Stash changes	`git stash` / `git stash pop`

12. Practical exercises (give to students)

1. Create a GitHub account and add your SSH key.
2. Fork a small repo or use the class repo; clone it locally.
3. Create a branch named `firstname-lastname-githubusername` and add a `hello.py` that prints your name. Commit, push, open a PR.
4. Practice a conflict: two students edit the same `README.md` line, open PRs, and resolve the conflict together in a controlled demo.

13. Instructor notes (grading / automation)

- We will use a GitHub Action that **auto-merges** correctly formatted PRs. Make sure your folder and branch names follow the exact naming rules.
- For grading: I will look at your folder in `main` after merging. Keep your PR description filled so I can see what you submitted.

14. FAQ (short)

Q: Can I push directly to `main`?
A: No — `main` is protected. Create a branch and open a PR.

Q: What if I accidentally edited another student's folder?
A: Revert the change or ask the instructor for help. Don't force-push to undo shared history.

Q: I can't resolve a conflict — what do I do?
A: Open a GitHub Issue in the repo with the title `Help - <YourName> - Merge conflict` and paste the conflict text or screenshots.

15. Links & resources (recommended)

- Git reference: `https://git-scm.com/docs`
- GitHub Docs: `https://docs.github.com/`
- Git cheatsheet (PDF): search `git cheat sheet`

End of guide

Powered by: your instructor