Team Git Workflow Guide

This guide explains a clear step-by-step workflow for working collaboratively with persistent personal branches.

Goal: Ensure everyone works independently without breaking main, always keeps their branch up-to-date with main, and merges their work back into main so everyone starts from the latest version.

---

0. Connecting to the GitHub repository (first time only)

Why: You need to clone the project to your local computer so you can work on it.

Steps: 1. Make sure the project owner has added you as a collaborator on GitHub. 2. On your computer, open a terminal or Git GUI. 3. Clone the repository:

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

4. Move into the repository folder:

```
cd repo
```

Now you're connected to the GitHub repository.

---

1. Initial Setup (once at the start)

Why: We create persistent personal branches so everyone has their own workspace and avoids constantly creating new branches.

Steps:

```
git checkout main              # Switch to main branch
git pull origin main          # Get latest main branch
git checkout -b yourname      # Create your branch, e.g. A, B, or C
git push -u origin yourname   # Push branch to remote so others can see it
```

Now each person has their own branch: `A`, `B`, `C`.

---

2. Starting a Coding Session

Why: We sync our branch with main to ensure we work with the latest version and avoid conflicts later.

Steps:

```
git checkout yourname         # Switch to your branch
git pull origin yourname      # Get your branch updates from remote
git fetch origin               # Get latest info from remote main branch
git merge origin/main          # Merge main into your branch
```

If conflicts appear: - Open conflicted file in your IDE. - Decide which version to keep or combine both. - Remove conflict markers:

```
<<<<<<< HEAD
your changes
=======
their changes
>>>>>>> origin/main
```

- Save file, then:

```
git add filename
git commit
```

---

3. Coding

Why: Small, frequent commits make merges easier and improve clarity.

Steps:

```
git add -A
git commit -m "Brief description of changes"
```

---

4. Finishing a Coding Session

Why: We merge our branch into main so others start from the latest version.

Steps:

```
git checkout main              # Switch to main
 git pull origin main           # Get latest changes from remote main
git merge yourname             # Merge your branch into main
git push origin main           # Push updated main to remote
```

Note: Merging your branch into main does not delete your branch — it remains available for further work. In this workflow, each person keeps their branch until the project ends.

---

5. After Merging

Why: Each person starts from the latest main version and continues working in their branch.

Steps: - Keep your branch for future sessions. - Next session, repeat Step 2.

Example Flow for A, B, and C: 1. **A**: Pulls main, merges into branch A, works, merges into main. 2. **B**: Pulls main (now contains A's work), merges into branch B, works, merges into main. 3. **C**: Pulls main (contains A and B's work), merges into branch C, works, merges into main.

✅ This workflow keeps work organized, minimizes conflicts, and makes collaboration easy even for people not fluent in Git.