Common Git Problems for New Team Members (and Fixes)

❌ Forgetting to Pull Latest Code Before Starting

Problem: They start working on outdated code, causing conflicts later.

Symptoms:

- "Why is my code missing after merge?"
- Merge conflicts after pushing

Fix:

```
git checkout main
git pull origin main
git checkout your-branch
git merge main
```

Tip: Always pull before starting work each day.

❌ Pushing Directly to `main`

Problem: Team member skips branching and pushes code directly to `main`, breaking the project.

Fix:

- **Use branches only**:

```
git checkout -b feature-login
```

- Project lead can protect `main` on GitHub (Settings > Branches > Protect main)

❌ Working on the Same File Together

Problem: Two people edit the same file (e.g. `index.html`), then push — leading to **merge conflicts**

Symptoms:

- Git says:

```
CONFLICT (content): Merge conflict in index.html
```

Fix:

- Git will mark the file like:

  ```
  <<<<<<< HEAD
  <h1>My code</h1>
  =======
  <h1>Their code</h1>
  >>>>>>> other-branch
  ```

- Manually edit, then:

```
git add .
git commit -m "Resolved conflict"
```

❌ Accidentally Committing Unwanted Files

Problem: Committing files like `.env`, `node_modules`, `.DS_Store`

Fix:

1. Use `.gitignore`:

   ```
   node_modules/
   .env
   *.log
   ```

2. Revert if needed:

   ```
   git rm --cached filename
   ```

❌ Too Many or Bad Commits

Problem: Messy commits like:

```
commit1: "aaaa"
commit2: "trying again"
commit3: "final fix"
```

Fix:

- Use meaningful commit messages:

```
git commit -m "Fix: Add validation for ticker input field"
```

- Optional: squash multiple commits during merge

❌ Forgot to Create a Branch

Problem: Starts working directly on `main` or wrong branch.

Fix:

```
git checkout -b my-new-feature
git push origin my-new-feature
```

❌ Merge Confusion

Problem: Unsure when to use `merge`, `pull`, or `rebase`

Tip:

- Beginners should stick with:

```
git pull origin main
git merge main
```

- Rebase is for advanced workflows

❌ Confused by Detached HEAD

Problem: They checkout a commit instead of a branch.

Fix:

```
git checkout main
```

Or create a new branch from there:

```
bash
CopyEdit
git checkout -b new-branch
```

❌ Not Syncing with Team After PR Merge

Problem: Their local code is behind after someone else merged a PR.

Fix:

```
git checkout main
git pull origin main
```

❌ Accidentally Deleted a File or Branch

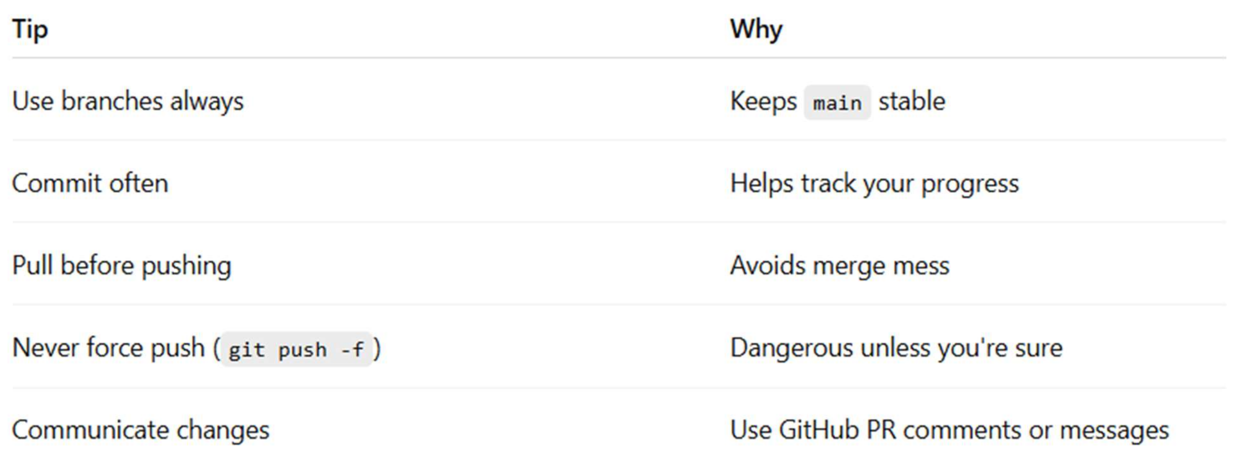
Fix for file:

```
git checkout HEAD -- filename
```

Fix for branch (if pushed before):

```
git checkout -b lost-branch origin/lost-branch
```

Final Tips for New Git Users



Tip	Why
Use branches always	Keeps `main` stable
Commit often	Helps track your progress
Pull before pushing	Avoids merge mess
Never force push (`git push -f`)	Dangerous unless you're sure
Communicate changes	Use GitHub PR comments or messages