About Git Conflicts

Common Mistakes Beginner Programmers Make (That Cause Git Conflicts)

1. Working on the same file at the same time

- Two people edit **the same lines** of the same file
- Git doesn't know which version is correct

2. Not pulling before starting work

- You start coding without:

```
git pull
```

- Your local code becomes **outdated**

3. Large commits with many changes

- Changing many files at once
- Harder for Git to auto-merge

4. Editing generated or config files unnecessarily

- Examples:
  - `package-lock.json`
  - `.env`
  - `yarn.lock`
- These files change easily and cause conflicts

5. Working directly on `main` / `master`

- Beginners often commit directly to main branch

- Increases chance of conflicts

6. Copy–paste code without checking latest version

- Code structure may have changed already

7. Not understanding what a conflict actually is

- Thinking conflict = error
- Actually, conflict = **Git asking you to decide**

Easy Steps to Reduce Git Conflicts (Best Practices)

1. Always pull before you start coding

```
git pull origin main
```

Do this **every time** before work

2. Use feature branches

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

Never work directly on `main`

3. Make small, focused commits

- One feature = one commit
- One bug fix = one commit

```
git commit -m "Add login validation"
```

4. Push regularly

- Don’t wait days to push
- Small pushes = fewer conflicts

5. Avoid editing the same file together

- Tell teammates:

> “I’m working on auth.controller.ts”

6. Use `.gitignore` properly

- Ignore:
  - `.env`
  - `node_modules`
  - build folders

7. Communicate with your team

- Simple message:

> “I’m updating user schema now”

What Is a Git Conflict? (Simple Explanation)

Git conflict means:

> ❝ Two changes happened in the same place, and Git doesn’t know which one to keep ❞

Git **stops merging** and waits for **you**.

How to Identify a Git Conflict

When you run:

```
git pull
```

or

```
git merge
```

You may see:

```
CONFLICT (content): Merge conflict in file.js
```

Git marks conflict like this 👇

```
<<<<<<< HEAD
your code
=======
their code
>>>>>>> feature-branch
```

Step-by-Step: How to Solve Git Conflict (Easy Way)

Step 1: Open the conflicted file

Look for:

```
<<<<<<<
=======
>>>>>>>
```

Step 2: Decide what to keep

You can:

- Keep **your code**
- Keep **their code**
- Combine both

Example:

```
// final decision
const user = await User.findById(id)
```

Remove ALL conflict markers:

```
<<<<<<< ======= >>>>>>>
```

Step 3: Save the file

Step 4: Tell Git conflict is solved

```
git add file.js
```

Step 5: Complete the merge

```
git commit
```

Conflict resolved!

Useful Commands for Conflict Handling

Command	Purpose
`git status`	See conflicted files
`git diff`	See differences
`git checkout --theirs file.js`	Keep their version
`git checkout --ours file.js`	Keep your version