Collaborative Git Workflow for Windows: Resolving Merge Conflicts

---

The previous exercise (Collaborative Git Workflow) we discussed how two developers can work on the same project using Git. However, what happens when both developers make different changes to the same part of the code? Git will raise a conflict, which the developers need to resolve manually. Here is how you can handle it:

1. **Identify the Conflict**: After attempting a `git merge` command, if Git detects a conflict, it will output a message indicating there's a merge conflict. You'll see something like this:

   ```
   CONFLICT (content): Merge conflict in file.txt
   Automatic merge failed; fix conflicts and then commit the result.
   ```

2. **Understand the Conflict**: Open the file with conflicts. You'll see the changes from the different branches marked clearly. Here's an example of what it may look like:

   ```
   <<<<<<< HEAD
   This is the line from your branch.
   =======
   This is the line from the branch you're merging.
   >>>>>>> branch-name
   ```

   The section between `<<<<<<< HEAD` and `=======` is your changes (the current branch). The section between `=======` and `>>>>>>> branch-name` contains the incoming changes from the branch you're merging.

3. **Resolve the Conflict**: Now, it's your job to decide which changes to keep. Edit the file to look exactly how you want it to look in the end - remove the conflict markers and decide which changes to keep.

4. **Stage the Resolved Conflict**: After you've resolved the conflict, use the `git add` command to stage the resolved file(s). For example:

```
git add file.txt
```

5. **Commit the Merge**: Once all conflicts are resolved, you can finish the merge by committing it. You can do this with the `git commit` command:

```
git commit -m "Resolved merge conflict for file.txt"
```

6. **Push the Changes**: After successfully resolving the conflict and committing the merge, push your changes to the remote repository:

```
git push origin your-branch-name
```

This way, you can resolve merge conflicts when working collaboratively on a project using Git. Always remember to communicate with your team when you encounter a merge conflict so you can decide the best course of action for your project.