**Git merge conflict**

-Version control systems are all about managing contributions between multiple distributed authors ( usually developers ).

-Sometimes multiple developers may try to edit the same content. If Developer A tries to edit code that Developer B is editing a conflict may occur.

-To alleviate the occurrence of conflicts developers will work in separate isolated branches.

**Understanding merge conflicts:**

* Conflicts generally arise when two people have changed the same lines in a file
* Or if one developer deleted a file while another developer was modifying it.
* In these cases, Git cannot automatically determine what is correct. Conflicts only affect the developer conducting the merge, the rest of the team is unaware of the conflict. Git will mark the file as being conflicted and halt the merging process.
* It is then the developers' responsibility to resolve the conflict.

**Resolving a merge conflict on GitHub:**

-You can resolve simple merge conflicts that involve competing line changes on GitHub, using the conflict editor.

-You can only resolve merge conflicts on GitHub that are caused by competing line changes, such as when people make different changes to the same line of the same file on different branches in your Git repository. For all other types of merge conflicts, you must resolve the conflict locally on the command line.

1. Under your repository name, click Pull requests.
2. In the "Pull Requests" list, click the pull request with a merge conflict that you'd like to resolve.
3. Near the bottom of your pull request, click Resolve conflicts.
4. Decide if you want to keep only your branch's changes, keep only the other branch's changes, or make a brand new change, which may incorporate changes from both branches. Delete the conflict markers <<<<<<<, =======, >>>>>>> and make the changes you want in the final merge.
5. If you have more than one merge conflict in your file, scroll down to the next set of conflict markers and repeat steps four and five to resolve your merge conflict.
6. Once you've resolved all the conflicts in the file, click Mark as resolved.
7. If you have more than one file with a conflict, select the next file you want to edit on the left side of the page under "conflicting files" and repeat steps four through seven until you've resolved all of your pull request's merge conflicts.
8. Once you've resolved all your merge conflicts, click Commit merge. This merges the entire base branch into your head branch.
9. If prompted, review the branch that you are committing to.
10. To merge your pull request, click Merge pull request. For more information about other pull request merge options, see "Merging a pull request."