**Merging Branches & Conflicts**

Merging is Git’s way of putting a forked history back together again. The git merge command lets you take the independent lines of development created by git branch and integrate them into a single branch.

Git merge will combine multiple sequences of commits into one unified history. In the most frequent use cases, git merge is used to combine two branches. The following examples in this document will focus on this branch merging pattern. In these scenarios, git merge takes two commit pointers, usually the branch tips, and will find a common base commit between them. Once Git finds a common base commit it will create a new “merge commit” that combines the changes of each queued merge commit sequence.

**Things You Should Know**

* Git merge will combine multiple sequences of commits into one unified history.
* In the most frequent use cases, git merge is used to combine two branches.
* Git merge takes commit pointers, usually the branch tips, and finds a common base commit between them.
* Once Git finds a common base commit it will create a new “merge commit” that combines the changes of each queued merge commit sequence.
* If the two branches you’re trying to merge both changed the same part of the same file, Git won’t be able to figure out which version to use.
  + When such a situation occurs, it stops right before the merge commit so that you can resolve the conflicts manually.
  + When you encounter a merge conflict, running the git status command shows you which files need to be resolved.