Branching & Merging in Git

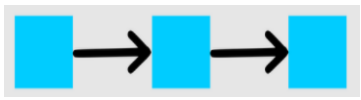
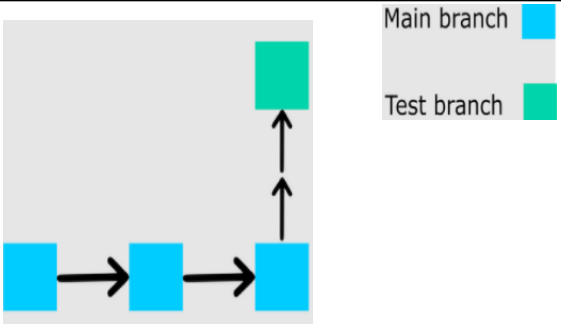
Topics covered:

- Git branching.
- Git merging.
    - Fast forward merge.
    - Recursive merge.
- How do merge conflicts occur in Git?
- How to resolve merge conflicts?
- Commands to resolve conflicts.
- Example - Merge conflict.

Topics in Detail:

Git Branching

- **Branching** helps in creating a copy of a file without messing up the original copy. We can either merge the changes or keep it independent.

Before Branching	After Branching
	

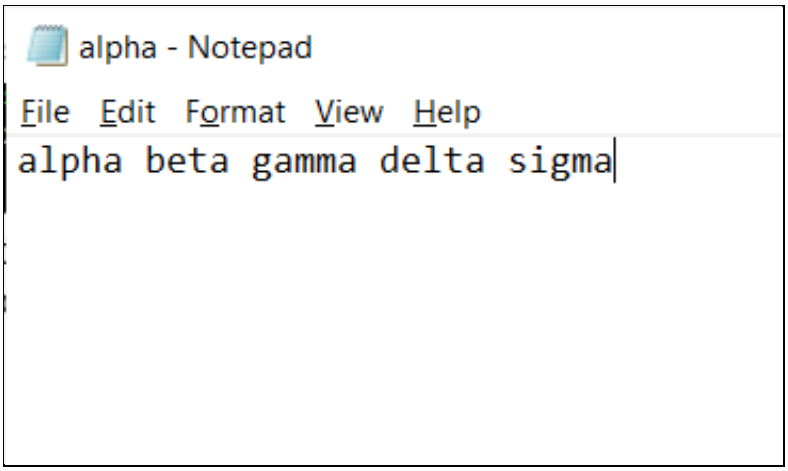
- I want to make changes in a file, but am not sure of changing the main file. So let's create a new branch named **test**.
- To create a new branch, use the following command.

```
$ git checkout -b branchname
```

    - **Checkout** - tells Git to switch to the branch.
    - **-b** - tells Git to create a new branch.

```
[redacted]@LAPTOP-S9VIU5AR MINGW64 ~/git_demo/Changes (master)
$ git checkout -b test
Switched to a new branch 'test'
```

- We can make some changes to the file. The branch has to be committed before merging these changes with the main branch.




```
[redacted]@LAPTOP-S9VIU5AR MINGW64 ~/git_demo/Changes (test)
$ git status
On branch test
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   alpha.txt

no changes added to commit (use "git add" and/or "git commit -a")

[redacted]@LAPTOP-S9VIU5AR MINGW64 ~/git_demo/Changes (test)
$ git add .

[redacted]@LAPTOP-S9VIU5AR MINGW64 ~/git_demo/Changes (test)
$ git commit -m "add branch"
[test 2f674a7] add branch
 1 file changed, 1 insertion(+), 1 deletion(-)
```

- To list all the branches existing in the repository, use the following command.

```
[redacted]@LAPTOP-S9VIU5AR MINGW64 ~/git_demo/Changes (test)
$ git branch
  master
* test
```

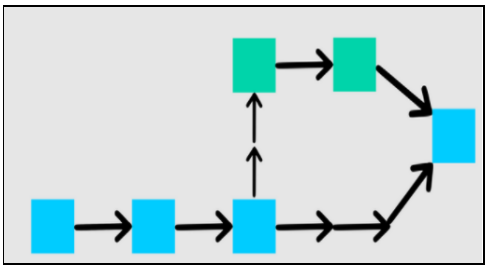
- The active branch is represented in green color.
- After committing the test branch, switch to the main branch by using the following command.

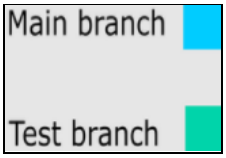
```
$ git checkout  main_branchname
```

```
[redacted]@LAPTOP-S9VIU5AR MINGW64 ~/git_demo/Changes (test)
$ git status
On branch test
nothing to commit, working tree clean

[redacted]@LAPTOP-S9VIU5AR MINGW64 ~/git_demo/Changes (test)
$ git checkout master
Switched to branch 'master'
Your branch is up to date with 'origin/master'.
```

Git Merging

- Created branches are to be merged with the main branch. So after merge, our repository will be like this





- Now, let's merge the test branch to the main branch using the following command.

```
$ git merge test
```

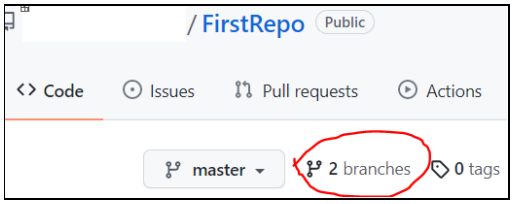
```
[redacted]@LAPTOP-S9VIU5AR MINGW64 ~/git_demo/Changes (master)
$ git merge test
Updating c1a53b1..2f674a7
Fast-forward
 alpha.txt | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)
```

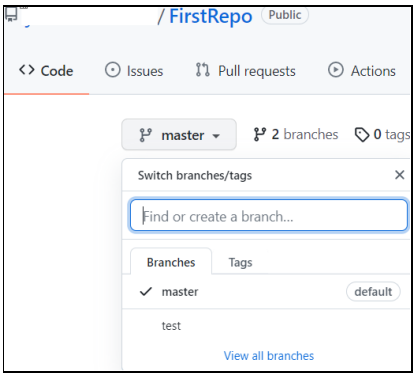
- If you want to push your branch to GitHub, then your active branch should test. For which first we have to checkout to test and then run the push command.

```
$ git checkout branchname
$ git push -u origin branchname
```

```
[redacted]@LAPTOP-S9VIU5AR MINGW64 ~/git_demo/Changes (master)
$ git checkout test
Switched to branch 'test'

[redacted]@LAPTOP-S9VIU5AR MINGW64 ~/git_demo/Changes (test)
$ git push -u origin test
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 293 bytes | 293.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'test' on GitHub by visiting:
remote:      https://github.com/[redacted]/FirstRepo/pull/new/test
remote:
To https://github.com/[redacted]/FirstRepo.git
 * [new branch]      test -> test
branch 'test' set up to track 'origin/test'.
```

- Refresh the GitHub page to view the changes.

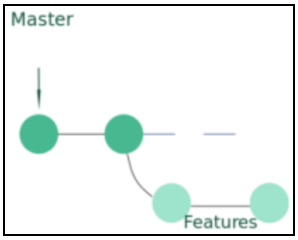




- Select the test branch in the above dropdown to view the changes made in the .txt file.

Fast Forward Merge

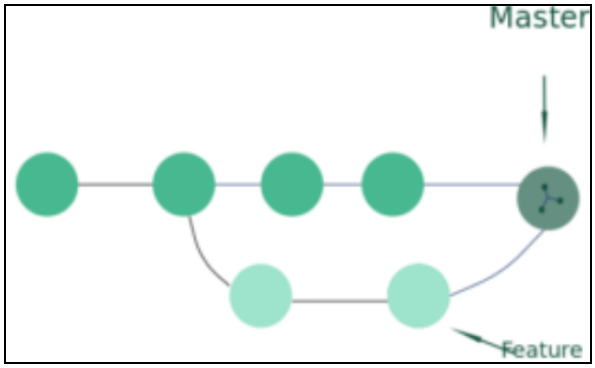
- Fast Forward Merge can occur only when there is a **linear path** from the current branch to the target branch.
- Actually, for this type of merge, Git has to **integrate the histories** forward from the current branch to the target branch.
- If the branches have **diverged,** then the Fast forward merge will **not be possible.**



- **Command**

Recursive Merge

- After branching and making some commits in the branch, there are some commits in the master as well.
- So at the time of the merge, the Git recurses over the branch and creates a new merge commit.
- Merge commit will have two parents.

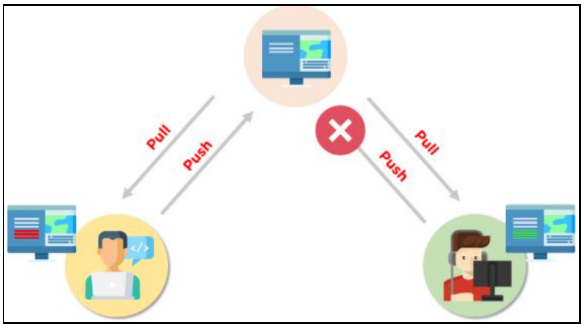


- **Command**

```
$ git merge - -no- ff
```

Git Merge Conflict

- Merge Conflict occurs when Git is **unable to resolve** the differences in code between **two commits**.
- When the commits are on **different lines,** then Git can **merge** the changes automatically.



- If two developers work in the same code and try to push the changes back to the repository, Conflict occurs. To avoid such conflicts, developers should work in separate branches.
- Git Merge command combines **separate branches** and resolves conflicts.

Occurrence of Merge conflicts:

- **Start of Merge process:**
  - Merge won't start when there are changes in the stage area of the working directory.
- **During the Merge process:**

- If there is a conflict between the local branch and the branch being merged, then it will fail during the merge process.
- In this case, the conflicted files are to be resolved manually.

How to resolve Merge Conflicts in Git?

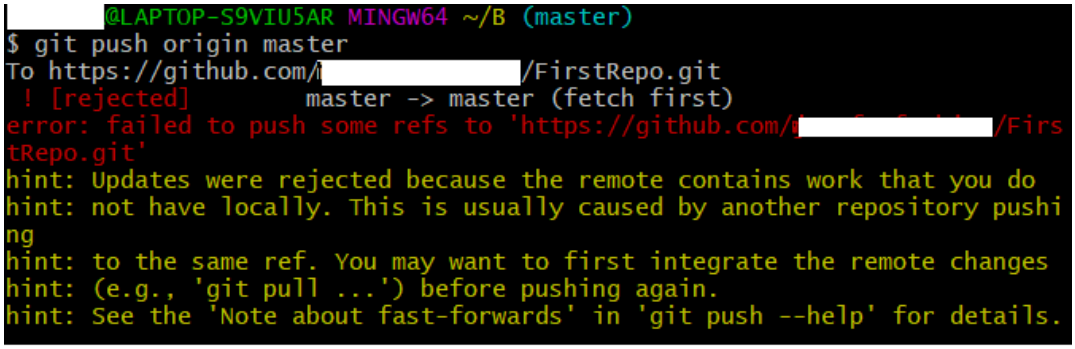
- Open the conflicted file and make the changes.
- After making changes, use the git **add command** to **stage** the new merged content.
- Create a new commit with the **git commit command**.
- Git will create a **new merge commit**.

Commands to resolve conflicts

- **git log –merge** - To **list** the conflict causing commits.
- **git diff** - To identify the difference between the states of repositories or files.
- **git checkout** - To **undo** the changes in the files or for changing the branches.
- **git reset –mixed** - To **undo** changes in the working directory and staging area.
- **git merge –abort** - To **exit** the merge process and return to the state before the merge began
- **git reset** - To **reset** the conflicted files to their original state.

Example: Merge Conflict

- Let’s create two local repositories A and B.
- Then pull the files to be cloned from GitHub in both the local repositories. Assuming that two developers work on the same file.
- Make appropriate changes in both repositories one by one.
- Now let’s try committing them one after the other and push them to the GitHub repository.
- While pushing the changes from the second repository, we will get the conflict as shown below.

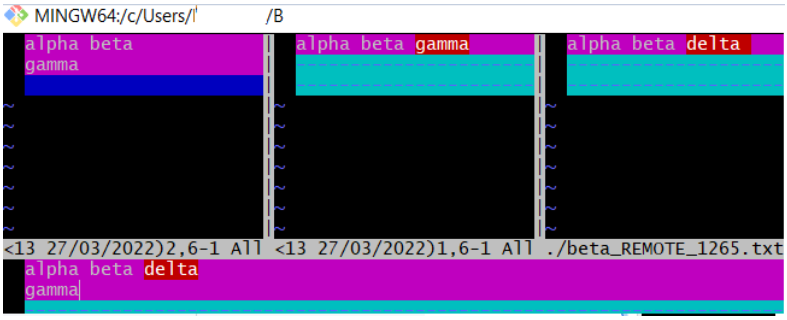


- As the conflict has occurred, we try to merge the conflict with the commands mentioned below

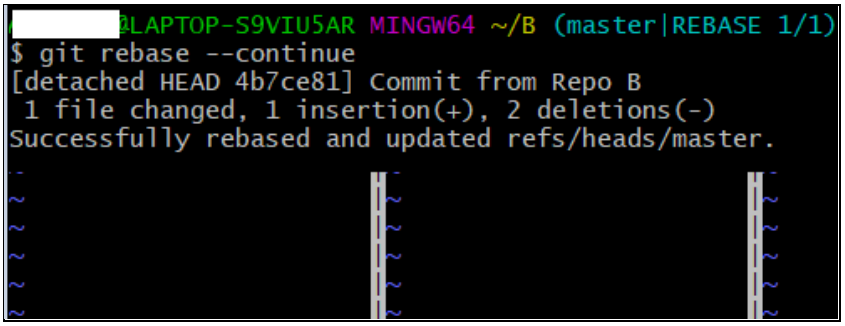
```
[redacted]@LAPTOP-S9VIU5AR MINGW64 ~/B (master)
$ git pull --rebase origin master
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 3 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 275 bytes | 5.00 KiB/s, done.
From https://github.com/[redacted]/FirstRepo
 * branch            master     -> FETCH_HEAD
   c1a53b1..cb77315  master     -> origin/master
Auto-merging beta.txt
CONFLICT (content): Merge conflict in beta.txt
error: could not apply 76d484d... Commit from Repo B
hint: Resolve all conflicts manually, mark them as resolved with
hint: "git add/rm <conflicted_files>", then run "git rebase --continue".
hint: You can instead skip this commit: run "git rebase --skip".
hint: To abort and get back to the state before "git rebase", run "git reb
ase --abort".
Could not apply 76d484d... Commit from Repo B

[redacted]@LAPTOP-S9VIU5AR MINGW64 ~/B (master|REBASE 1/1)
$ git mergetool

This message is displayed because 'merge.tool' is not configured.
See 'git mergetool --tool-help' or 'git help config' for more details.
'git mergetool' will now attempt to use one of the following tools:
opendiff kdiff3 tkdiff xxdiff meld tortoisemerge gvimdiff diffuse diffmerg
e ecmerge p4merge araxis bc codecompare smerge emerge vimdiff nvimdiff
Merging:
beta.txt

Normal merge conflict for 'beta.txt':
  {local}: modified file
  {remote}: modified file
Hit return to start merge resolution tool (vimdiff): return
4 files to edit
```

- Following this, a comparison page will open

- This compares the difference in code with repository A, remote repository, and repository B.
- Let's try resolving the conflicts manually, as shown below.



- After resolving the conflict manually we can merge the files using the **git rebase –continue** command and at last, we can push the file into the GitHub repository.



- We can refresh the GitHub repository and check the conflicts that occurred and the corresponding changes made.

- If we click on the Commits link, it will list all the commits done so far. We can also view all the changes made to the file.