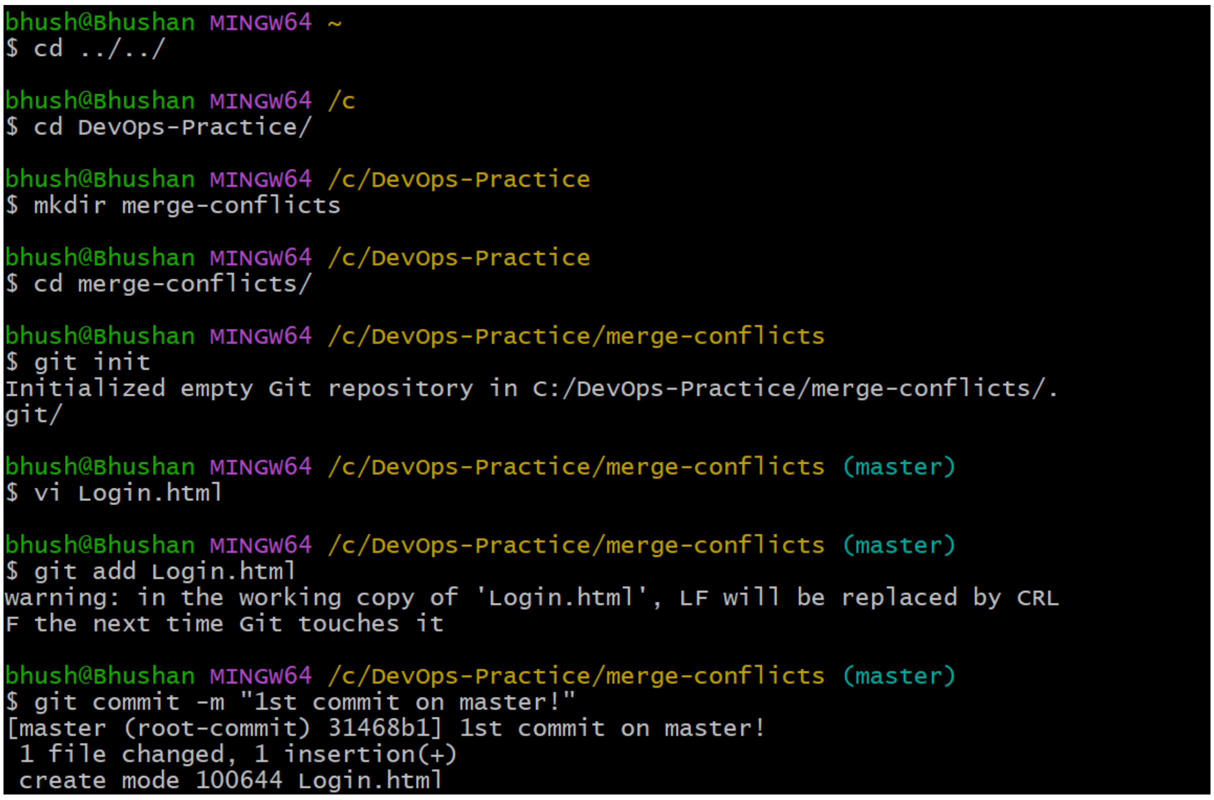
Assignment no 4

Merge Conflicts

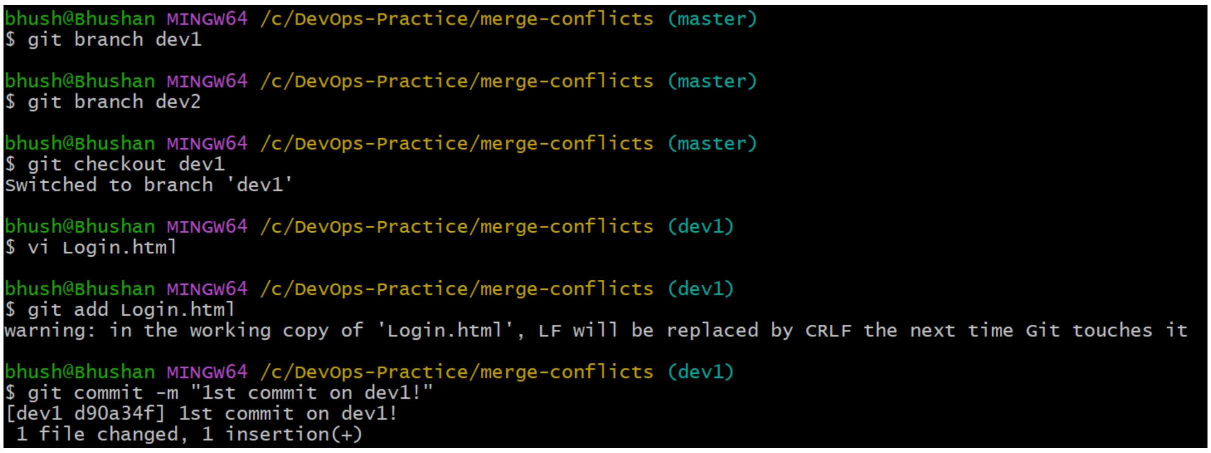
1) Create a use case to generate merge conflicts and resolve it.

Point 1: Make directory having name merge-conflict and initialize it. After initialization create a file having name login.html. Stage and commit it.



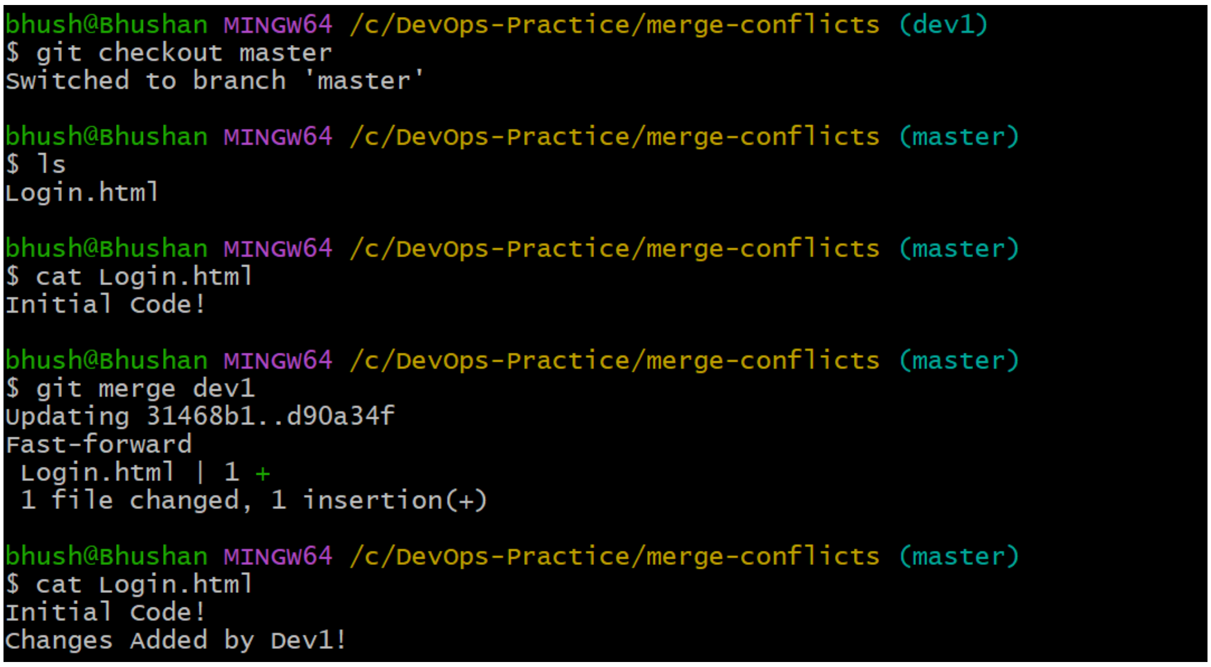
```
bhush@Bhushan MINGW64 ~
$ cd ../../

bhush@Bhushan MINGW64 /c
$ cd DevOps-Practice/

bhush@Bhushan MINGW64 /c/DevOps-Practice
$ mkdir merge-conflicts

bhush@Bhushan MINGW64 /c/DevOps-Practice
$ cd merge-conflicts/

bhush@Bhushan MINGW64 /c/DevOps-Practice/merge-conflicts
$ git init
Initialized empty Git repository in C:/DevOps-Practice/merge-conflicts/.
git/

bhush@Bhushan MINGW64 /c/DevOps-Practice/merge-conflicts (master)
$ vi Login.html

bhush@Bhushan MINGW64 /c/DevOps-Practice/merge-conflicts (master)
$ git add Login.html
warning: in the working copy of 'Login.html', LF will be replaced by CRL
F the next time Git touches it

bhush@Bhushan MINGW64 /c/DevOps-Practice/merge-conflicts (master)
$ git commit -m "1st commit on master!"
[master (root-commit) 31468b1] 1st commit on master!
 1 file changed, 1 insertion(+)
 create mode 100644 Login.html
```

Point 2: Create 2 branches dev1 and dev2. Checkout to dev1 and edit the login.html on dev1. Stage it and commit it.



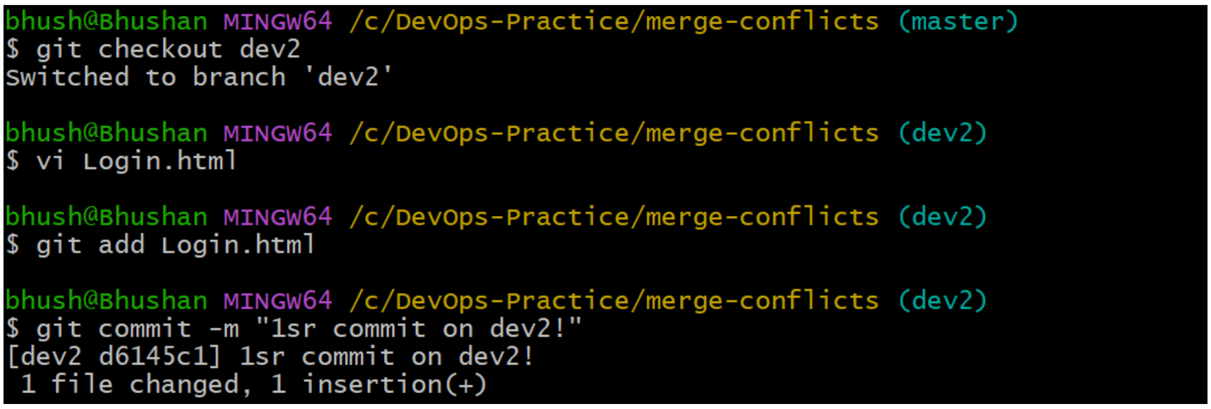
```
bhush@Bhushan MINGW64 /c/DevOps-Practice/merge-conflicts (master)
$ git branch dev1

bhush@Bhushan MINGW64 /c/DevOps-Practice/merge-conflicts (master)
$ git branch dev2

bhush@Bhushan MINGW64 /c/DevOps-Practice/merge-conflicts (master)
$ git checkout dev1
Switched to branch 'dev1'

bhush@Bhushan MINGW64 /c/DevOps-Practice/merge-conflicts (dev1)
$ vi Login.html

bhush@Bhushan MINGW64 /c/DevOps-Practice/merge-conflicts (dev1)
$ git add Login.html
warning: in the working copy of 'Login.html', LF will be replaced by CRLF the next time Git touches it

bhush@Bhushan MINGW64 /c/DevOps-Practice/merge-conflicts (dev1)
$ git commit -m "1st commit on dev1!"
[dev1 d90a34f] 1st commit on dev1!
 1 file changed, 1 insertion(+)
```

Point 3: Now checkout to master and check content of the file. It is observed that content is present up to first commit. So to reflect the changes from dev1 to master use git merge command. Now Check the content. Login.html is now updated with all commits until now.



```
bhush@Bhushan MINGW64 /c/DevOps-Practice/merge-conflicts (dev1)
$ git checkout master
Switched to branch 'master'

bhush@Bhushan MINGW64 /c/DevOps-Practice/merge-conflicts (master)
$ ls
Login.html

bhush@Bhushan MINGW64 /c/DevOps-Practice/merge-conflicts (master)
$ cat Login.html
Initial Code!

bhush@Bhushan MINGW64 /c/DevOps-Practice/merge-conflicts (master)
$ git merge dev1
Updating 31468b1..d90a34f
Fast-forward
 Login.html | 1 +
 1 file changed, 1 insertion(+)

bhush@Bhushan MINGW64 /c/DevOps-Practice/merge-conflicts (master)
$ cat Login.html
Initial Code!
Changes Added by Dev1!
```

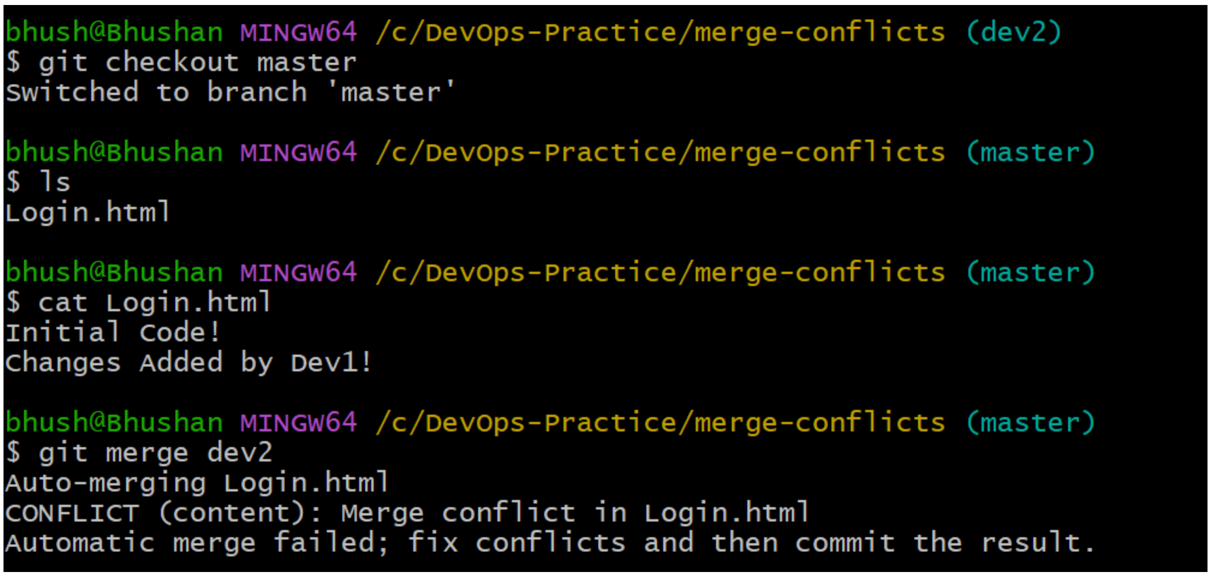
Point 4: Now checkout to dev2 , edit the login.html file, stage and commit it.



```
bhush@Bhushan MINGW64 /c/DevOps-Practice/merge-conflicts (master)
$ git checkout dev2
Switched to branch 'dev2'

bhush@Bhushan MINGW64 /c/DevOps-Practice/merge-conflicts (dev2)
$ vi Login.html

bhush@Bhushan MINGW64 /c/DevOps-Practice/merge-conflicts (dev2)
$ git add Login.html

bhush@Bhushan MINGW64 /c/DevOps-Practice/merge-conflicts (dev2)
$ git commit -m "1sr commit on dev2!"
[dev2 d6145c1] 1sr commit on dev2!
 1 file changed, 1 insertion(+)
```

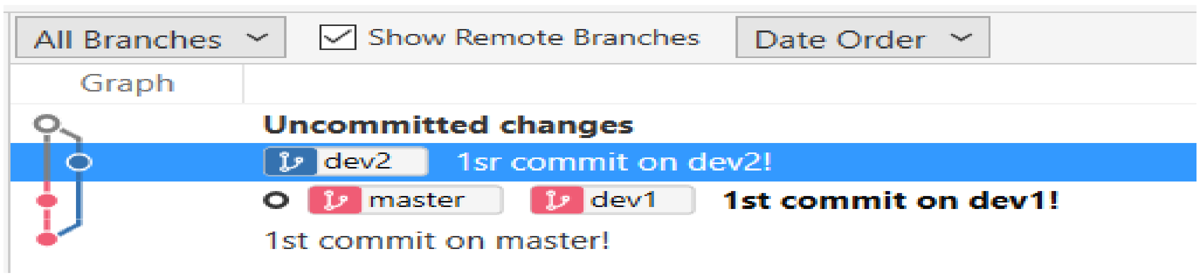
Point 5: Now checkout the master and check the content of the login.html file. Here observe that dev1 changes only added in login.html. So to add the dev2 changes in the login.html file merge the changes from dev2 to master.

Disclaimer / Note :- But here we get merge conflict

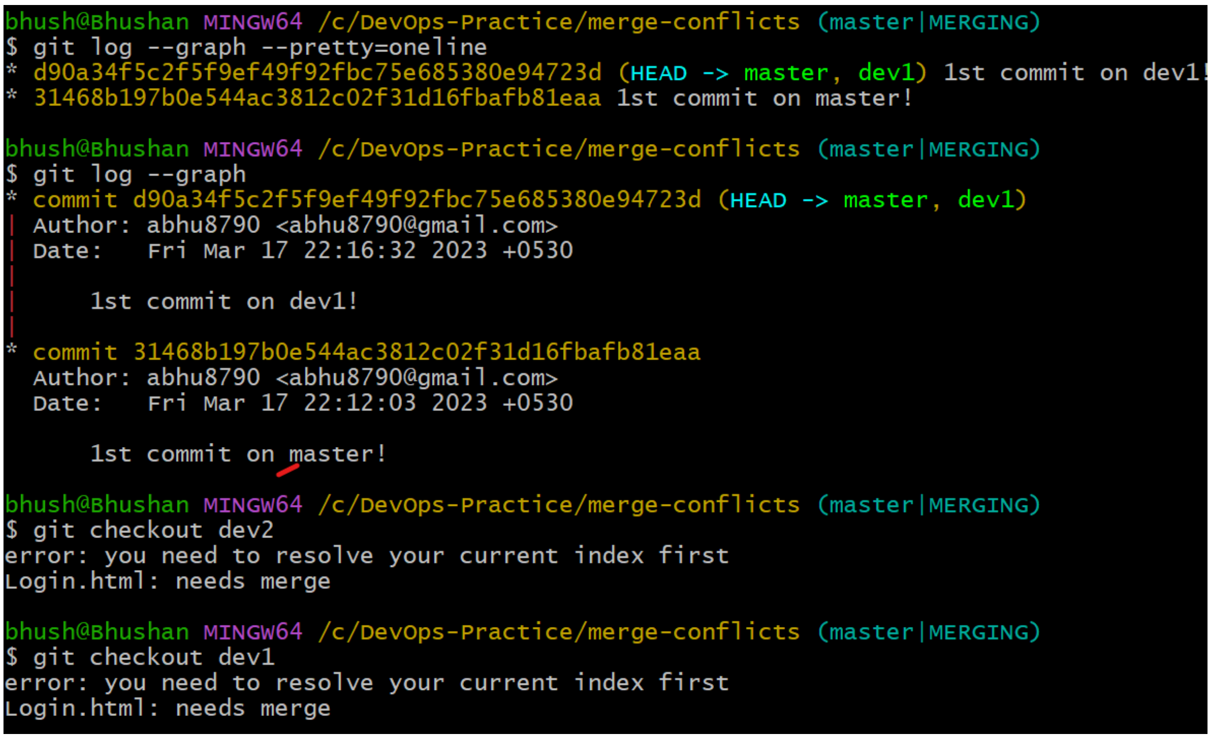


```
bhush@Bhushan MINGW64 /c/DevOps-Practice/merge-conflicts (dev2)
$ git checkout master
Switched to branch 'master'

bhush@Bhushan MINGW64 /c/DevOps-Practice/merge-conflicts (master)
$ ls
Login.html

bhush@Bhushan MINGW64 /c/DevOps-Practice/merge-conflicts (master)
$ cat Login.html
Initial Code!
Changes Added by Dev1!

bhush@Bhushan MINGW64 /c/DevOps-Practice/merge-conflicts (master)
$ git merge dev2
Auto-merging Login.html
CONFLICT (content): Merge conflict in Login.html
Automatic merge failed; fix conflicts and then commit the result.
```

Until Now Source diagram of the commits are as follows: -

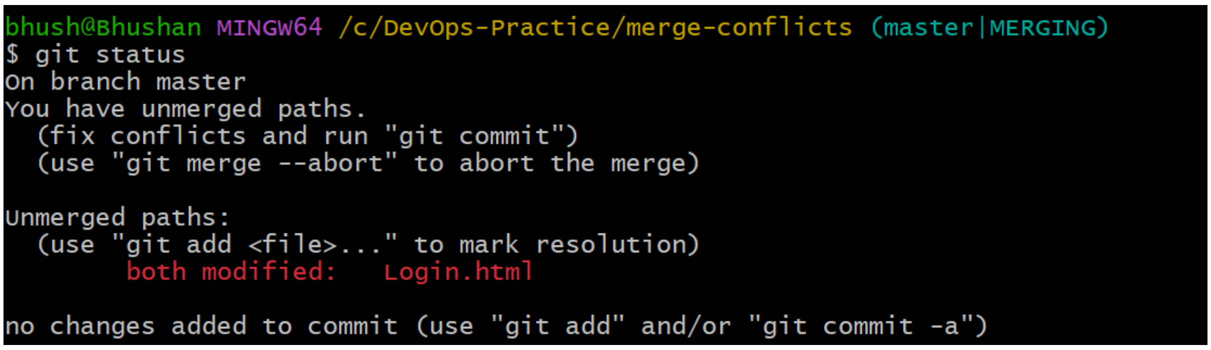


If we check the graphical representation we observe that HEAD is point towards the master and dev1 because we merged it but when we trying to checkout to another branch it shows mergin



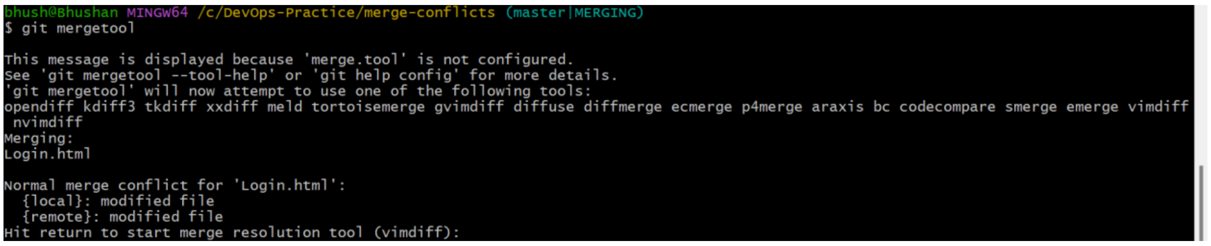
```
bhush@Bhushan MINGW64 /c/DevOps-Practice/merge-conflicts (master|MERGING)
$ git log --graph --pretty=oneline
* d90a34f5c2f5f9ef49f92fbc75e685380e94723d (HEAD -> master, dev1) 1st commit on dev1!
* 31468b197b0e544ac3812c02f31d16fbafb81eaa 1st commit on master!

bhush@Bhushan MINGW64 /c/DevOps-Practice/merge-conflicts (master|MERGING)
$ git log --graph
* commit d90a34f5c2f5f9ef49f92fbc75e685380e94723d (HEAD -> master, dev1)
| Author: abhu8790 <abhu8790@gmail.com>
| Date:   Fri Mar 17 22:16:32 2023 +0530
|
|     1st commit on dev1!
|
* commit 31468b197b0e544ac3812c02f31d16fbafb81eaa
  Author: abhu8790 <abhu8790@gmail.com>
  Date:   Fri Mar 17 22:12:03 2023 +0530

      1st commit on master!

bhush@Bhushan MINGW64 /c/DevOps-Practice/merge-conflicts (master|MERGING)
$ git checkout dev2
error: you need to resolve your current index first
Login.html: needs merge

bhush@Bhushan MINGW64 /c/DevOps-Practice/merge-conflicts (master|MERGING)
$ git checkout dev1
error: you need to resolve your current index first
Login.html: needs merge
```

Point 6: If we check the status then it shows that both modified. It means that dev1 and dev2 changed the login.html file. But have merge conflicts.

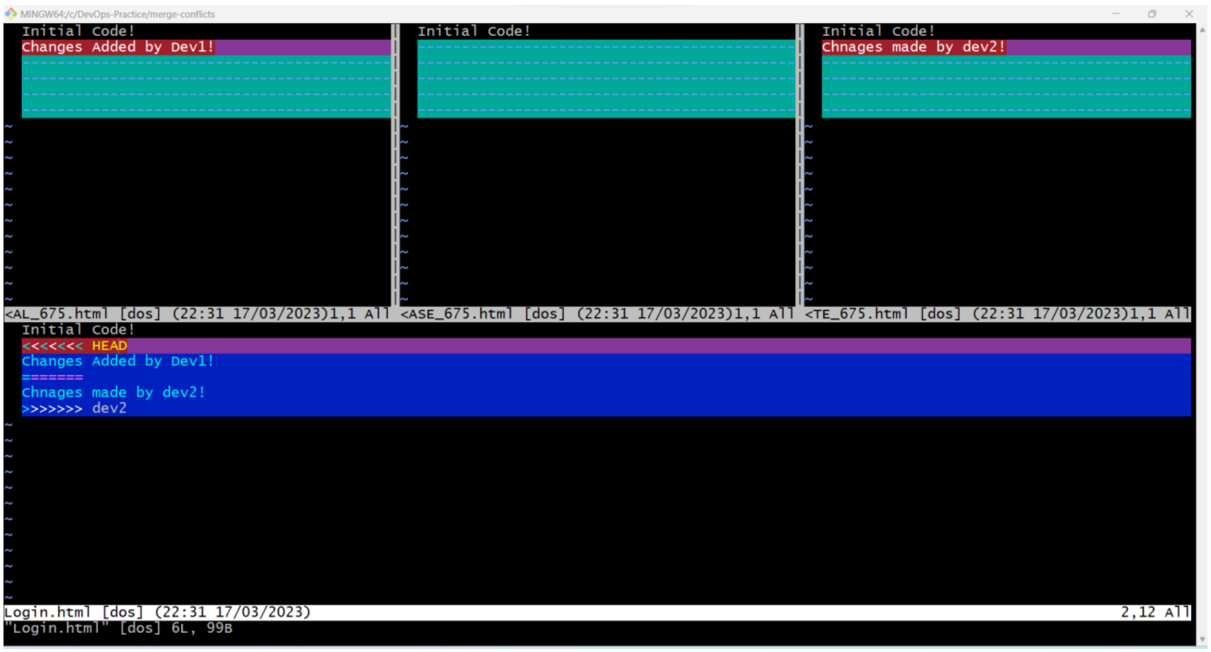


```
bhush@Bhushan MINGW64 /c/DevOps-Practice/merge-conflicts (master|MERGING)
$ git status
On branch master
You have unmerged paths.
  (fix conflicts and run "git commit")
  (use "git merge --abort" to abort the merge)

Unmerged paths:
  (use "git add <file>..." to mark resolution)
        both modified:   Login.html

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

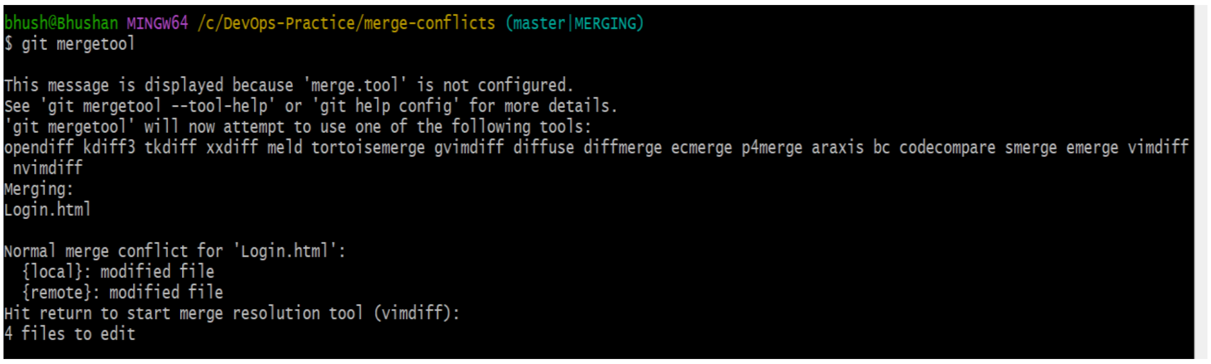
Point 7: If we want to solve the merge conflicts then we use mergetool. Hit Enter it will shows the changes all branches. Edit it and save it.



```
bhush@Bhushan MINGW64 /c/DevOps-Practice/merge-conflicts (master|MERGING)
$ 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 diffmerge ecmerge p4merge araxis bc codecompare smerge emerge vimdiff
 nvimdiff
Merging:
Login.html

Normal merge conflict for 'Login.html':
  {local}: modified file
  {remote}: modified file
Hit return to start merge resolution tool (vimdiff):
```

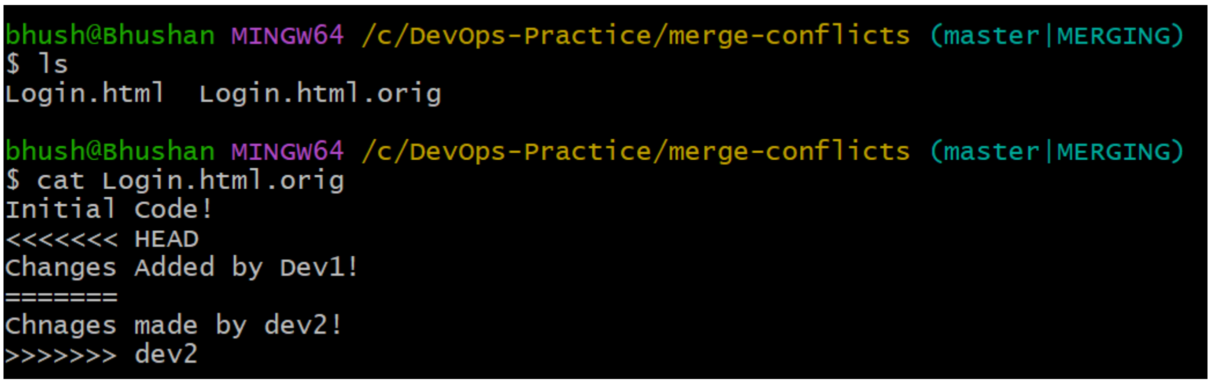


```
MINGW64:/c/DevOps-Practice/merge-conflicts
  Initial Code!                          Initial Code!                          Initial Code!
  Changes Added by Dev1!                                                        Chnages made by dev2!
<AL_675.html [dos] (22:31 17/03/2023)1,1 All <ASE_675.html [dos] (22:31 17/03/2023)1,1 All <TE_675.html [dos] (22:31 17/03/2023)1,1 All
  Initial Code!
  <<<<<<< HEAD
  Changes Added by Dev1!
  =======
  Chnages made by dev2!
  >>>>>>> dev2
Login.html [dos] (22:31 17/03/2023)                                                                        2,12 All
"Login.html" [dos] 6L, 99B
```



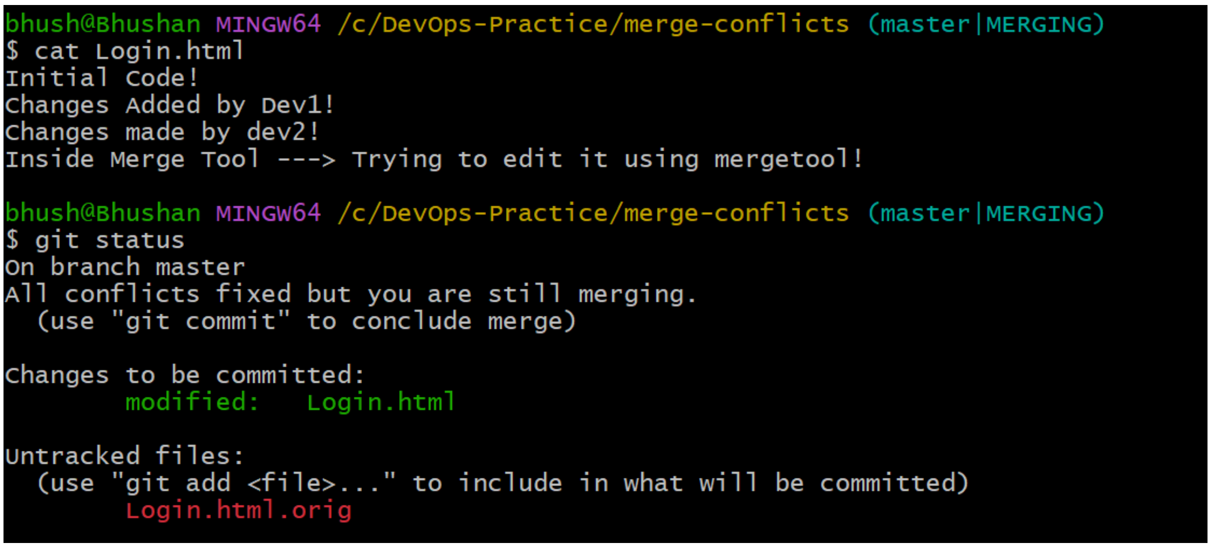
```
bhush@Bhushan MINGW64 /c/DevOps-Practice/merge-conflicts (master|MERGING)
$ git mergetool

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 diffmerge ecmerge p4merge araxis bc codecompare smerge emerge vimdiff
 nvimdiff
Merging:
Login.html

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

Point 8: If we check the list of files in local directory then we get login.htm and login.html.orig files. If we check the content of the login.html.orig , it is observed that it display content exists in merging branch and current branch.



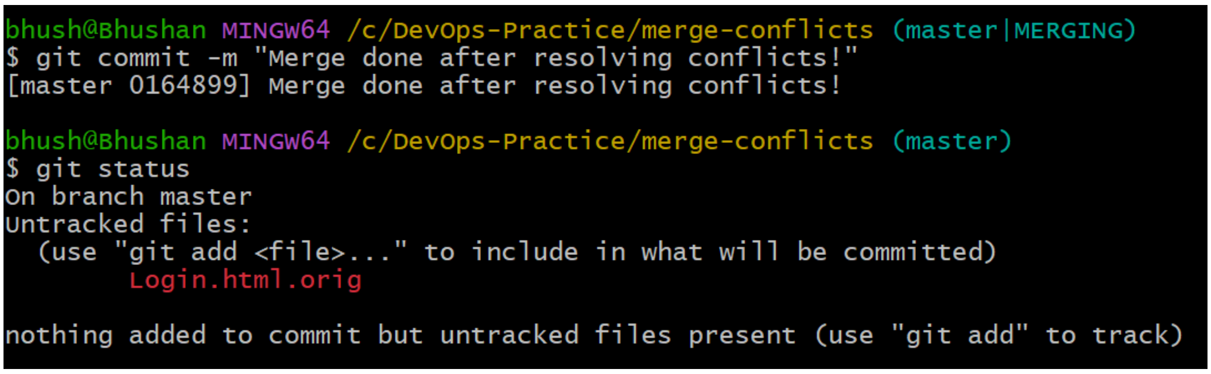
```
bhush@Bhushan MINGW64 /c/DevOps-Practice/merge-conflicts (master|MERGING)
$ ls
Login.html  Login.html.orig

bhush@Bhushan MINGW64 /c/DevOps-Practice/merge-conflicts (master|MERGING)
$ cat Login.html.orig
Initial Code!
<<<<<<< HEAD
Changes Added by Dev1!
=======
Chnages made by dev2!
>>>>>>> dev2
```

Point 9: Check the content of login.html , this is the file that contain all the changes that we made after resolving the conflicts. If we check status then it says that login.html is modified and login.html.orig is untracked. This untracked file is used for our reference.



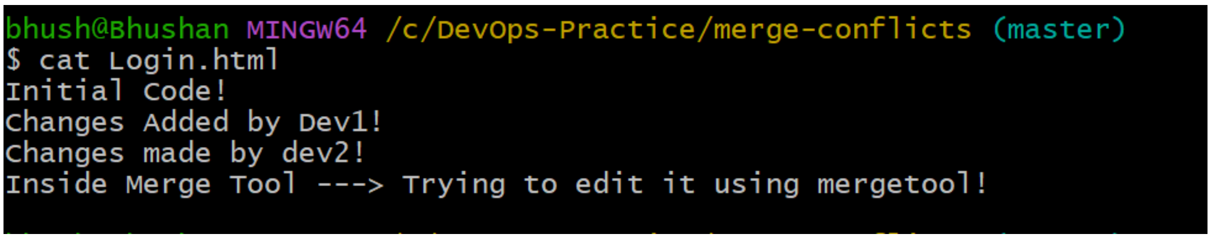
```
bhush@Bhushan MINGW64 /c/DevOps-Practice/merge-conflicts (master|MERGING)
$ cat Login.html
Initial Code!
Changes Added by Dev1!
Changes made by dev2!
Inside Merge Tool ---> Trying to edit it using mergetool!

bhush@Bhushan MINGW64 /c/DevOps-Practice/merge-conflicts (master|MERGING)
$ git status
On branch master
All conflicts fixed but you are still merging.
  (use "git commit" to conclude merge)

Changes to be committed:
        modified:   Login.html

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        Login.html.orig
```

Point 10: Now we want to commit those changes to master branch then commit it with message. If we check the status the only untrack file are there that is login.html.orig



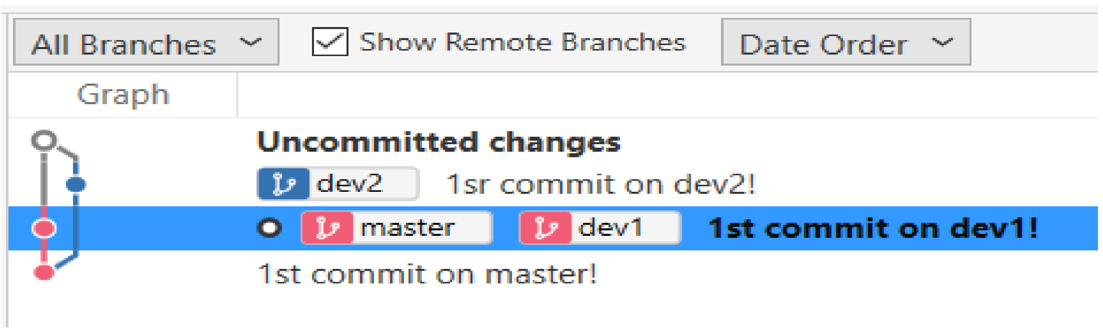
```
bhush@Bhushan MINGW64 /c/DevOps-Practice/merge-conflicts (master|MERGING)
$ git commit -m "Merge done after resolving conflicts!"
[master 0164899] Merge done after resolving conflicts!

bhush@Bhushan MINGW64 /c/DevOps-Practice/merge-conflicts (master)
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        Login.html.orig

nothing added to commit but untracked files present (use "git add" to track)
```

Point 11: Check the content of the login.html file we have all the changes in that file.



```
bhush@Bhushan MINGW64 /c/DevOps-Practice/merge-conflicts (master)
$ cat Login.html
Initial Code!
Changes Added by Dev1!
Changes made by dev2!
Inside Merge Tool ---> Trying to edit it using mergetool!
```

Final Diagram after solving merge conflicts :



Uncommitted changes
dev2 1sr commit on dev2!
master dev1 1st commit on dev1!
1st commit on master!