

NAME: Ronak Suthar

Roll. no.: 39

Department: CSBS (Computer Science and Business)

Year: Final

Assignment 2

EXERCISE 1:

Main Task

1. Create a new directory and change into it.

```
C:\Users\User>mkdir Assignment2Dir  
C:\Users\User>cd Assignment2Dir
```

2. Use the init command to create a Git repository in that directory.

```
C:\Users\User\Assignment2Dir>git init  
Initialized empty Git repository in C:/Users/User/Assignment2Dir/.git/
```

3. Observe that there is now a .git directory.

```
C:\Users\User\Assignment2Dir>dir /a:h  
Volume in drive C is Windows 11  
Volume Serial Number is D230-3700  
  
Directory of C:\Users\User\Assignment2Dir  
  
12-07-2024  05.57 PM    <DIR>          .git  
                0 File(s)                0 bytes  
                1 Dir(s)  110,832,627,712 bytes free  
  
C:\Users\User\Assignment2Dir>|
```

4. Create a README file.

```
Git CMD
C:\Users\User\Assignment2Dir>dir /a
Volume in drive C is Windows 11
Volume Serial Number is D230-3700

Directory of C:\Users\User\Assignment2Dir

12-07-2024  06.01 PM    <DIR>          .
12-07-2024  05.57 PM    <DIR>          ..
12-07-2024  05.57 PM    <DIR>          .git
12-07-2024  06.01 PM                12 README.md
               1 File(s)                12 bytes
               3 Dir(s) 110,831,788,032 bytes free

C:\Users\User\Assignment2Dir>
```

5. Look at the output of the status command; the README you created should appear as an untracked file.

```
Git CMD
C:\Users\User\Assignment2Dir>git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
      README.md

nothing added to commit but untracked files present (use "git add" to track)
C:\Users\User\Assignment2Dir>
```

6. Use the add command to add the new file to the staging area. Again, look at the output of the status command.

```
C:\Users\User\Assignment2Dir>git add .

C:\Users\User\Assignment2Dir>git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   README.md

C:\Users\User\Assignment2Dir>
```

7. Now use the commit command to commit the contents of the staging area.

```
C:\Users\User\Assignment2Dir>git commit -m "Readme added"
[master (root-commit) e99eb56] Readme added
 1 file changed, 1 insertion(+)
 create mode 100644 README.md

C:\Users\User\Assignment2Dir>
```

8. Create a src directory and add a couple of files to it.

```
Git CMD
C:\Users\User\Assignment2Dir>tree /f
Folder PATH listing for volume Windows 11
Volume serial number is D230-3700
C:.
|
|_ README.md
|_ src
    |_ file1.txt
    |_ file2.txt
    |_ file3.txt

C:\Users\User\Assignment2Dir>
```

9. Use the add command, but name the directory, not the individual files. Use the status command. See how both files have been staged. Commit them.

```
Git CMD
C:\Users\User\Assignment2Dir>git add src

C:\Users\User\Assignment2Dir>git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   src/file1.txt
    new file:   src/file2.txt
    new file:   src/file3.txt

C:\Users\User\Assignment2Dir>git commit -m "added src directoru"
[master c7eb560] added src directoru
 3 files changed, 3 insertions(+)
 create mode 100644 src/file1.txt
 create mode 100644 src/file2.txt
 create mode 100644 src/file3.txt

C:\Users\User\Assignment2Dir>
```

10. Make a change to one of the files. Use the diff command to view the details of the change.

```
C:\Users\User\Assignment2Dir>git diff src/file1.txt src/file1.txt
diff --git a/src/file1.txt b/src/file1.txt
index e8f5d6a..03cf565 100644
--- a/src/file1.txt
+++ b/src/file1.txt
@@ -1,1 @@
-"file one"
+"updated text in file one"

C:\Users\User\Assignment2Dir>
```

11. Next, add the changed file, and notice how it moves to the staging area in the status output. Also observe that the diff command you did before using add now gives no output. Why not? What do you have to do to see a diff of the things in the staging area? (Hint: review the slides if you can't remember.)

>> git diff --staged

```
C:\Users\User\Assignment2Dir>echo "file changes in file3" >> file4.txt
C:\Users\User\Assignment2Dir>git add file4.txt
C:\Users\User\Assignment2Dir>git diff file4.txt file4.txt

C:\Users\User\Assignment2Dir>git diff --staged file4.txt file4.txt
diff --git a/file4.txt b/file4.txt
index e351e09..6ca112e 100644
--- a/file4.txt
+++ b/file4.txt
@@ -1,2 @@
 "file 4 form barnch 1"
+"file changes in file3"

C:\Users\User\Assignment2Dir>
```

```

C:\Users\User\Assignment2Dir>cd src
C:\Users\User\Assignment2Dir\src>git add file1.txt
C:\Users\User\Assignment2Dir\src>cd ..
C:\Users\User\Assignment2Dir>git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   src/file1.txt

C:\Users\User\Assignment2Dir>

```

12. Now – without committing – make another change to the same file you changed in step 10. Look at the status output, and the diff output. Notice how you can have both staged and unstaged changes, even when you're talking about a single file. Observe the difference when you use the add command to stage the latest round of changes. Finally, commit them. You should now have started to get a feel for the staging area.

```

Git CMD
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   src/file1.txt

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:   src/file1.txt

C:\Users\User\Assignment2Dir>git diff src/file1.txt src/file1.txt
diff --git a/src/file1.txt b/src/file1.txt
index 03cf565..f5af91d 100644
--- a/src/file1.txt
+++ b/src/file1.txt
@@ -1,1 @@
-"updated text in file one"
+"changes from step 12"

C:\Users\User\Assignment2Dir>

```

```
C:\Users\User\Assignment2Dir>git add src/file1.txt

C:\Users\User\Assignment2Dir>git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   src/file1.txt

C:\Users\User\Assignment2Dir>git commit -m "step 12 commit"
[master 099fc51] step 12 commit
1 file changed, 1 insertion(+), 1 deletion(-)
```

13. Use the log command in order to see all of the commits you made so far.

```
C:\Users\User\Assignment2Dir>git log
commit 099fc51f3381451e2f8d98e2d466fdddebc510bf (HEAD -> master)
Author: Ronak <ronakronu02@gmail.com>
Date:   Fri Jul 12 18:18:36 2024 +0530

    step 12 commit

commit c7eb560793d6ce5d01300d2f519ef7ec8dc66044
Author: Ronak <ronakronu02@gmail.com>
Date:   Fri Jul 12 18:11:17 2024 +0530

    added src directory

commit e99eb56aeaf940b59feb82cf274a8e9133702c5d
Author: Ronak <ronakronu02@gmail.com>
Date:   Fri Jul 12 18:03:55 2024 +0530

    Readme added
```

14. Use the show command to look at an individual commit. How many characters of the commit identifier can you get away with typing at a minimum?

One commit hash character:

```
C:\Users\User\Assignment2Dir>git show 0
fatal: ambiguous argument '0': unknown revision or path not in the working tree.
Use '--' to separate paths from revisions, like this:
'git <command> [<revision>...] -- [<file>...]
```

Two commit hash character:

```
C:\Users\User\Assignment2Dir>git show 01
fatal: ambiguous argument '01': unknown revision or path not in the working tree.
Use '--' to separate paths from revisions, like this:
'git <command> [<revision>...] -- [<file>...]
```

four commit hash character: (minimum number of characters to show)

```
C:\Users\User\Assignment2Dir>git show 099f
commit 099fc51f3381451e2f8d98e2d466fdddebc510bf (HEAD -> master)
Author: Ronak <ronakronu02@gmail.com>
Date:   Fri Jul 12 18:18:36 2024 +0530

    step 12 commit

diff --git a/src/file1.txt b/src/file1.txt
index e8f5d6a..f5af91d 100644
--- a/src/file1.txt
```

15. Make a couple more commits, at least one of which should add an extra file.

```
Git CMD
C:\Users\User\Assignment2Dir>git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    src/file4.txt

nothing added to commit but untracked files present (use "git add" to track)
C:\Users\User\Assignment2Dir>
```



```

C:\Users\User\Assignment2Dir>git log
commit 4fc5233b60b9ef108018ef7af1d410e6251ba8ba (HEAD -> master)
Author: Ronak <ronakronu02@gmail.com>
Date:   Fri Jul 12 18:24:39 2024 +0530

    extra file added

commit 099fc51f3381451e2f8d98e2d466fdddebc510bf
Author: Ronak <ronakronu02@gmail.com>
Date:   Fri Jul 12 18:18:36 2024 +0530

    step 12 commit

commit c7eb560793d6ce5d01300d2f519ef7ec8dc66044
Author: Ronak <ronakronu02@gmail.com>
Date:   Fri Jul 12 18:11:17 2024 +0530

    added src directoru

commit e99eb56aeaf940b59feb82cf274a8e9133702c5d
Author: Ronak <ronakronu02@gmail.com>
Date:   Fri Jul 12 18:03:55 2024 +0530

    Readme added

```

Stretch Task

1. Use the Git rm command to remove a file. Look at the status afterwards. Now commit the deletion.

```

C:\Users\User\Assignment2Dir\src>git rm file4.txt
rm 'src/file4.txt'

C:\Users\User\Assignment2Dir\src>dir
Volume in drive C is Windows 11
Volume Serial Number is D230-3700

Directory of C:\Users\User\Assignment2Dir\src

12-07-2024  06.26 PM    <DIR>          .
12-07-2024  06.04 PM    <DIR>          ..
12-07-2024  06.16 PM                25 file1.txt
12-07-2024  06.08 PM                13 file2.txt
12-07-2024  06.08 PM                15 file3.txt
               3 File(s)                53 bytes
               2 Dir(s)  110,815,244,288 bytes free

C:\Users\User\Assignment2Dir\src>

```

2. Delete another file, but this time do not use Git to do it; e.g. if you are on Linux, just use the normal (non-Git) rm command; on Windows use del.

```
C:\Users\User\Assignment2Dir\src>del file3.txt

C:\Users\User\Assignment2Dir\src>dir
Volume in drive C is Windows 11
Volume Serial Number is D230-3700

Directory of C:\Users\User\Assignment2Dir\src

12-07-2024  06.26 PM    <DIR>          .
12-07-2024  06.04 PM    <DIR>          ..
12-07-2024  06.16 PM                25 file1.txt
12-07-2024  06.08 PM                13 file2.txt
                2 File(s)                38 bytes
                2 Dir(s) 110,814,797,824 bytes free

C:\Users\User\Assignment2Dir\src>
```

3. Look at the status. Compare it to the status output you had after using the Git built-in rm command. Is anything different? After this, commit the deletion.

```
Git CMD
C:\Users\User\Assignment2Dir>git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        deleted:    src/file4.txt

Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        deleted:    src/file3.txt

C:\Users\User\Assignment2Dir>
C:\Users\User\Assignment2Dir>git add .

C:\Users\User\Assignment2Dir>git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        deleted:    src/file3.txt
        deleted:    src/file4.txt

C:\Users\User\Assignment2Dir>git commit -m "deleted file3 and file4"
[master 95b7942] deleted file3 and file4
 2 files changed, 2 deletions(-)
 delete mode 100644 src/file3.txt
 delete mode 100644 src/file4.txt
```

4. Use the Git mv command to move or rename a file; for example, rename README to README.txt. Look at the status. Commit the change.

```
C:\Users\User\Assignment2Dir>git mv README.md README.txt
```

```
C:\Users\User\Assignment2Dir>dir
Volume in drive C is Windows 11
Volume Serial Number is D230-3700
```

```
Directory of C:\Users\User\Assignment2Dir
```

```
12-07-2024  06.29 PM    <DIR>          .
12-07-2024  06.24 PM    <DIR>          ..
12-07-2024  06.01 PM                12 README.txt
12-07-2024  06.26 PM    <DIR>          src
                        1 File(s)        12 bytes
                        3 Dir(s)  110,817,923,072 bytes free
```

```
C:\Users\User\Assignment2Dir>git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        renamed:    README.md -> README.txt
```

```
C:\Users\User\Assignment2Dir>git commit -m "ex:1 stretch task step four commit"
[master 38398bf] ex:1 stretch task step four commit
 1 file changed, 0 insertions(+), 0 deletions(-)
 rename README.md => README.txt (100%)
```

```
C:\Users\User\Assignment2Dir>git log
commit 38398bffe20558ae8790121d1f334de2f17c2ea8 (HEAD -> master)
Author: Ronak <ronakronu02@gmail.com>
Date:   Fri Jul 12 18:30:40 2024 +0530

    ex:1 stretch task step four commit
```

5. Now do another rename, but this time using the operating system's command to do so. How does the status look? Will you get the right outcome if you were to commit at this point? (Answer: almost certainly not, so don't. 9) Work out how to get the status to show that it will not lose the file, and then commit. Did Git at any point work out that you had done a rename?

```

Directory of C:\Users\User\Assignment2Dir

12-07-2024  06.29 PM    <DIR>          .
12-07-2024  06.30 PM    <DIR>          ..
12-07-2024  06.01 PM                12 README.txt
12-07-2024  06.26 PM    <DIR>          src
                        1 File(s)          12 bytes
                        3 Dir(s)  110,815,657,984 bytes free

C:\Users\User\Assignment2Dir>ren README.txt README.md

C:\Users\User\Assignment2Dir>dir
Volume in drive C is Windows 11
Volume Serial Number is D230-3700

```

```

C:\Users\User\Assignment2Dir>git status
On branch master
Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        deleted:    README.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        README.md

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

```

```

C:\Users\User\Assignment2Dir>git commit -m "renamed readme.txt to readme.md \ren/"
[master 66d9a8a] renamed readme.txt to readme.md \ren/
 1 file changed, 1 insertion(+)
 create mode 100644 README.md

C:\Users\User\Assignment2Dir>git log
commit 66d9a8af6d1b147e0dadd45b027a774a71d99063 (HEAD -> master)
Author: Ronak <ronakronu02@gmail.com>
Date:   Fri Jul 12 18:40:20 2024 +0530

    renamed readme.txt to readme.md \ren/

```

6. Use git help log to find out how to get Git to display just the most recent 3 commits. Try it.

Note that these are applied before commit ordering and formatting options, such as `--reverse`.

`--<number>`

`-n <number>`

`--max-count=<number>`

Limit the number of commits to output.

```
C:\Users\User\Assignment2Dir>git log --max-count=3
commit 66d9a8af6d1b147e0dadd45b027a774a71d99063 (HEAD -> master)
Author: Ronak <ronakronu02@gmail.com>
Date:   Fri Jul 12 18:40:20 2024 +0530

    renamed readme.txt to readme.md \ren/

commit 38398bffe20558ae8790121d1f334de2f17c2ea8
Author: Ronak <ronakronu02@gmail.com>
Date:   Fri Jul 12 18:30:40 2024 +0530

    ex:1 stretch task step four commit

commit 95b79424197a17473643d17bedb6462f5e93685d
Author: Ronak <ronakronu02@gmail.com>
Date:   Fri Jul 12 18:28:24 2024 +0530

    deleted file3 and file4
```

7. If you don't remember, look back in the slides to see what the `--stat` option did on the `diff` command. Find out if this also works with the `show` command. How about the `log` command?

```
C:\Users\User\Assignment2Dir>git diff --stat 66d9a8af6d1b147e0dadd45b027a774a71d99063 38398bffe20558ae879
0121d1f334de2f17c2ea8
 README.md | 1 -
 1 file changed, 1 deletion(-)
```

```

C:\Users\User\Assignment2Dir>git log --stat
commit 66d9a8af6d1b147e0dadd45b027a774a71d99063 (HEAD -> master)
Author: Ronak <ronakronu02@gmail.com>
Date:   Fri Jul 12 18:40:20 2024 +0530

    renamed readme.txt to readme.md \ren/

README.md | 1 +
1 file changed, 1 insertion(+)

commit 38398bffe20558ae8790121d1f334de2f17c2ea8
Author: Ronak <ronakronu02@gmail.com>
Date:   Fri Jul 12 18:30:40 2024 +0530

    ex:1 stretch task step four commit

README.md => README.txt | 0
1 file changed, 0 insertions(+), 0 deletions(-)

commit 95b79424197a17473643d17bedb6462f5e93685d
Author: Ronak <ronakronu02@gmail.com>
Date:   Fri Jul 12 18:28:24 2024 +0530

    deleted file3 and file4

src/file3.txt | 1 -
src/file4.txt | 1 -
2 files changed, 2 deletions(-)

commit 4fc5233b60b9ef108018ef7af1d410e6251ba8ba
Author: Ronak <ronakronu02@gmail.com>
Date:   Fri Jul 12 18:24:39 2024 +0530

    extra file added

src/file4.txt | 1 +
1 file changed, 1 insertion(+)

```

8. Imagine you want to see a diff that summarizes all that happened between two commit identifiers. Use the diff command, specifying two commit identifiers joined by two dots (that is, something like abc123..def456). Check the output is what you expect.

```

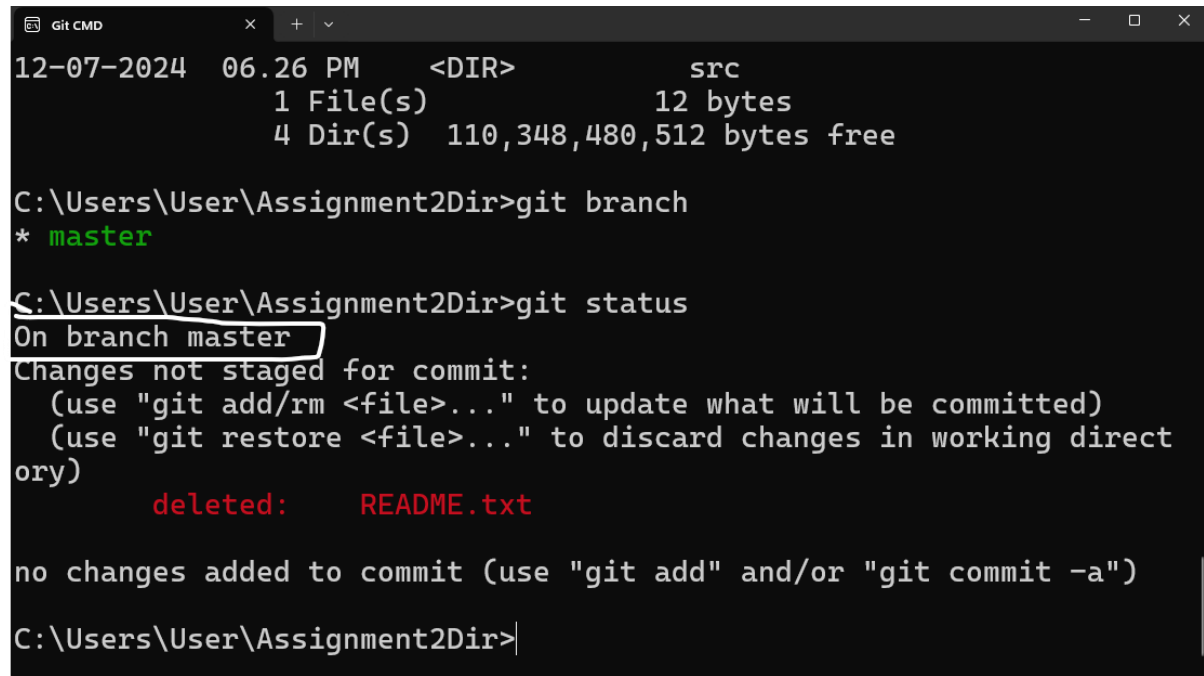
C:\Users\User\Assignment2Dir>git diff 66d9a8af6d1b147e0dadd45b027a774a71d99063..38398bffe20558ae8790121d1f334de2f17c2ea8
diff --git a/README.md b/README.md
deleted file mode 100644
index 5f3079e..0000000
--- a/README.md
+++ /dev/null
@@ -1,0,0 @@
- "# Hello"

```

Exercise 2

Main Task

Run the status command. Notice how it tells you what branch you are in.



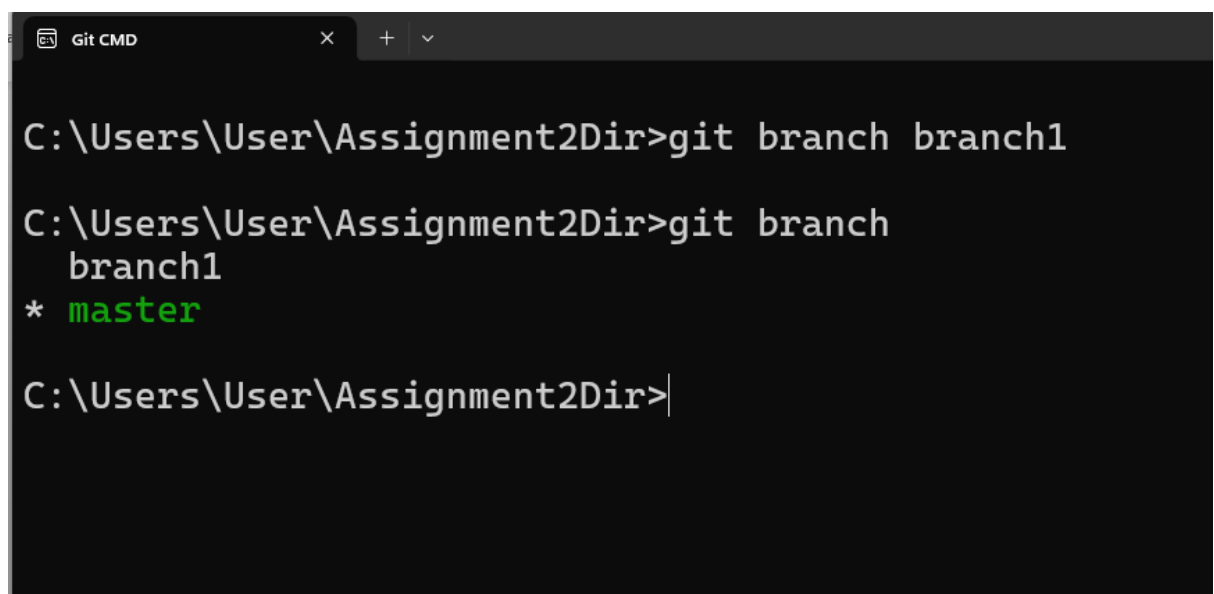
```
Git CMD
12-07-2024 06.26 PM <DIR> src
1 File(s) 12 bytes
4 Dir(s) 110,348,480,512 bytes free

C:\Users\User\Assignment2Dir>git branch
* master

C:\Users\User\Assignment2Dir>git status
On branch master
Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        deleted:    README.txt

no changes added to commit (use "git add" and/or "git commit -a")
C:\Users\User\Assignment2Dir>
```

Use the branch command to create a new branch.



```
Git CMD

C:\Users\User\Assignment2Dir>git branch branch1

C:\Users\User\Assignment2Dir>git branch
branch1
* master

C:\Users\User\Assignment2Dir>
```

3. Use the checkout command to switch to it.


```
C:\Users\User\Assignment2Dir>git checkout branch1
D      README.txt
Switched to branch 'branch1'

C:\Users\User\Assignment2Dir>git branch
* branch1
  master

C:\Users\User\Assignment2Dir>
```

4. Make a couple of commits in the branch – perhaps adding a new file and/or editing existing ones.

```
C:\Users\User\Assignment2Dir>dir
Volume in drive C is Windows 11
Volume Serial Number is D230-3700

Directory of C:\Users\User\Assignment2Dir

13-07-2024  03.03 PM    <DIR>          .
13-07-2024  03.02 PM    <DIR>          ..
13-07-2024  03.02 PM                19 branch2file.txt
13-07-2024  03.03 PM                19 branch2Javafile.java
12-07-2024  06.01 PM                12 README.md
12-07-2024  06.26 PM    <DIR>          src
                                3 File(s)          50 bytes
                                3 Dir(s)  110,345,605,120 bytes free

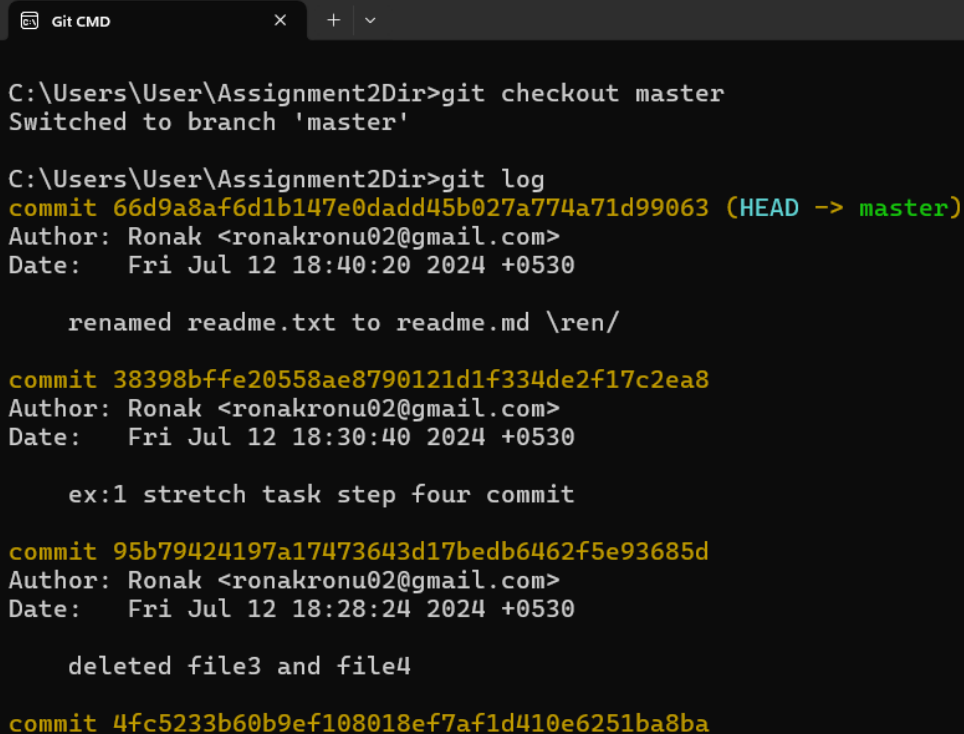
C:\Users\User\Assignment2Dir>
```

5. Use the log command to see the latest commits. The two you just made should be at the top of the list.

```
C:\Users\User\Assignment2Dir>git log
commit 99a8c9a29ffce7d95da7cc426f6aab8140ca1f86 (HEAD -> branch1)
Author: Ronak <ronakronu02@gmail.com>
Date:   Sat Jul 13 15:04:40 2024 +0530

    Branch 2 commit
```

6. Use the checkout command to switch back to the master branch. Run log again. Notice



```
Git CMD
C:\Users\User\Assignment2Dir>git checkout master
Switched to branch 'master'

C:\Users\User\Assignment2Dir>git log
commit 66d9a8af6d1b147e0dadd45b027a774a71d99063 (HEAD -> master)
Author: Ronak <ronakronu02@gmail.com>
Date:   Fri Jul 12 18:40:20 2024 +0530

    renamed readme.txt to readme.md \ren/

commit 38398bffe20558ae8790121d1f334de2f17c2ea8
Author: Ronak <ronakronu02@gmail.com>
Date:   Fri Jul 12 18:30:40 2024 +0530

    ex:1 stretch task step four commit

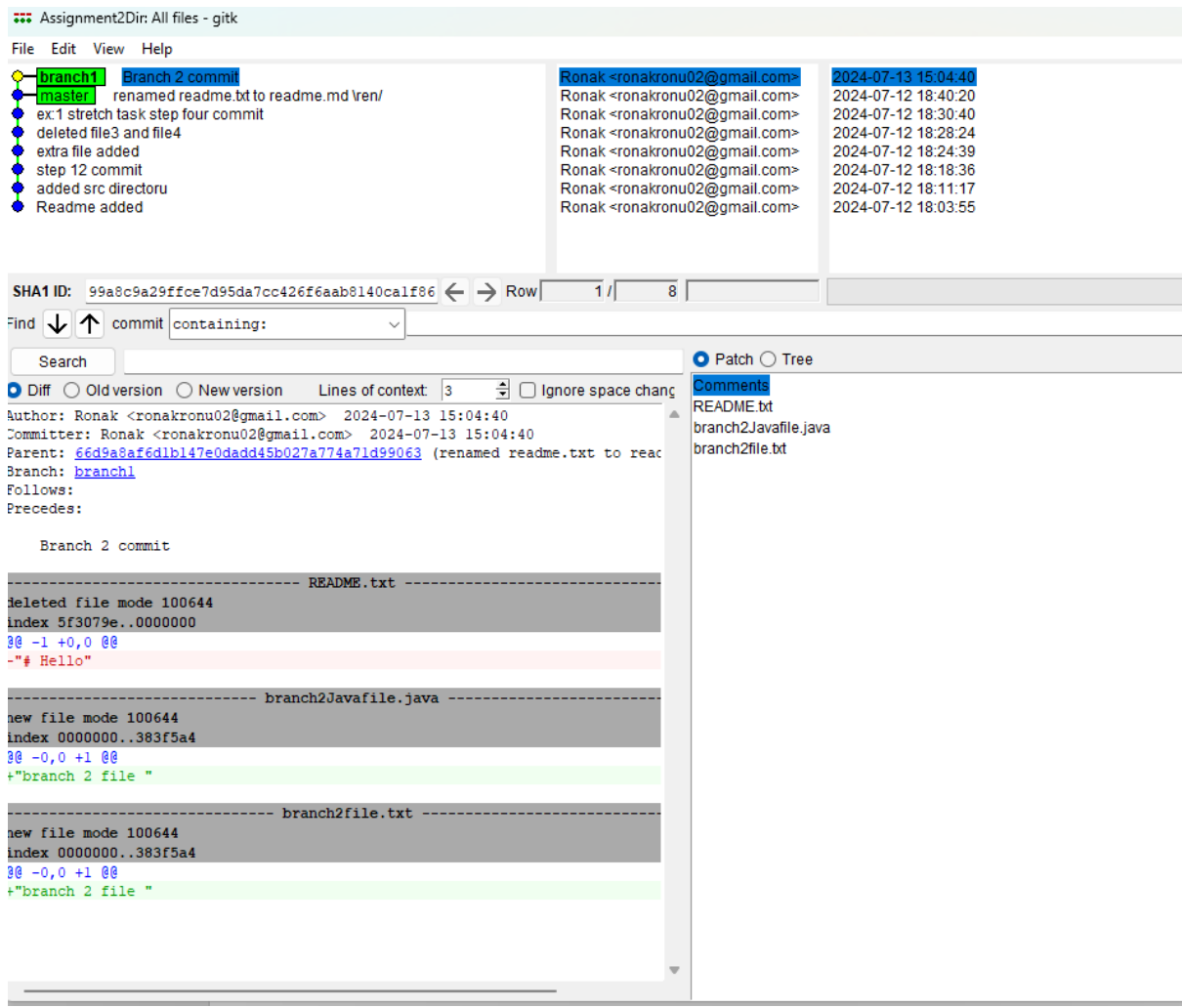
commit 95b79424197a17473643d17bedb6462f5e93685d
Author: Ronak <ronakronu02@gmail.com>
Date:   Fri Jul 12 18:28:24 2024 +0530

    deleted file3 and file4

commit 4fc5233b60b9ef108018ef7af1d410e6251ba8ba
```

your commits don't show up now. Check the files also – they should have their original contents.

7. Use the checkout command to switch back to your branch. Use gitk to take a look at the commit graph; notice it's linear.



8. Now checkout the master branch again. Use the merge command to merge your branch in to it. Look for information about it having been a fast-forward merge. Look at git log, and see that there is no merge commit. Take a look in gitk and see how the DAG is linear.

```
C:\Users\User\Assignment2Dir>git branch
```

```
branch1
```

```
* master
```

```
C:\Users\User\Assignment2Dir>git merge branch1
```

```
Updating 66d9a8a..99a8c9a
```

```
Fast-forward
```

```
README.txt | 1 -
```

```
branch2Javafile.java | 1 +
```

```
branch2file.txt | 1 +
```

```
3 files changed, 2 insertions(+), 1 deletion(-)
```

```
delete mode 100644 README.txt
```

```
create mode 100644 branch2Javafile.java
```

```
create mode 100644 branch2file.txt
```

```
C:\Users\User\Assignment2Dir>
```

```
C:\Users\User\Assignment2Dir>git log
```

```
commit 99a8c9a29ffce7d95da7cc426f6aab8140ca1f86 (HEAD -> master, branch1)
```

```
Author: Ronak <ronakronu02@gmail.com>
```

```
Date: Sat Jul 13 15:04:40 2024 +0530
```

```
Branch 2 commit
```

```
commit 66d9a8af6d1b147e0dadd45b027a774a71d99063
```

```
Author: Ronak <ronakronu02@gmail.com>
```

```
Date: Fri Jul 12 18:40:20 2024 +0530
```

```
renamed readme.txt to readme.md \ren/
```

```
commit 38398bffe20558ae8790121d1f334de2f17c2ea8
```

```
Author: Ronak <ronakronu02@gmail.com>
```

```
Date: Fri Jul 12 18:30:40 2024 +0530
```

```
ex:1 stretch task step four commit
```

```
commit 95b79424197a17473643d17bedb6462f5e93685d
```

```
Author: Ronak <ronakronu02@gmail.com>
```

```
Date: Fri Jul 12 18:28:24 2024 +0530
```

Assignment2Dir: All files - gitk

File Edit View Help

branch1 master Branch 2 commit

- renamed readme.txt to readme.md \ren/
- ex:1 stretch task step four commit
- deleted file3 and file4
- extra file added
- step 12 commit
- added src directoru
- Readme added

Ronak <ronakronu02@gmail.com> 2024-07-13 15:04:40

Ronak <ronakronu02@gmail.com> 2024-07-12 18:40:20

Ronak <ronakronu02@gmail.com> 2024-07-12 18:30:40

Ronak <ronakronu02@gmail.com> 2024-07-12 18:28:24

Ronak <ronakronu02@gmail.com> 2024-07-12 18:24:39

Ronak <ronakronu02@gmail.com> 2024-07-12 18:18:36

Ronak <ronakronu02@gmail.com> 2024-07-12 18:11:17

Ronak <ronakronu02@gmail.com> 2024-07-12 18:03:55

SHA1 ID: 99a8c9a29ffce7d95da7cc426f6aab8140ca1f86

Find commit containing:

Search

Diff Old version New version Lines of context: 3 Ignore space change

Author: Ronak <ronakronu02@gmail.com> 2024-07-13 15:04:40

Committer: Ronak <ronakronu02@gmail.com> 2024-07-13 15:04:40

Parent: 66d9a8af6d1b147e0dadd45b027a774a71d99063 (renamed readme.txt to readme.md)

Branches: branch1, master

Follows:

Precedes:

Branch 2 commit

deleted file mode 100644

index 5f3079e..0000000

@@ -1 +0,0 @@

-"# Hello"

branch2Javafile.java

new file mode 100644

index 0000000..383f5a4

@@ -0,0 +1 @@

+"branch 2 file "

branch2file.txt

new file mode 100644

index 0000000..383f5a4

@@ -0,0 +1 @@

+"branch 2 file "

Comments

README.txt

branch2Javafile.java

branch2file.txt

9. Switch back to your branch. Make a couple more commits.

```
C:\Users\User\Assignment2Dir>git status
On branch branch1
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    newfilecpp.cpp
    newfiledoc.doc
    newfilejava.java

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

C:\Users\User\Assignment2Dir>git add .

C:\Users\User\Assignment2Dir>git commit -m "added three file to branch2 "
[branch1 59bd778] added three file to branch2
3 files changed, 3 insertions(+)
create mode 100644 newfilecpp.cpp
create mode 100644 newfiledoc.doc
create mode 100644 newfilejava.java
```

10. Switch back to master. Make a commit there, which should edit a different file from the

ones you touched in your branch – to be sure there is no conflict.

```
C:\Users\User\Assignment2Dir>git add newfiledoc.doc

C:\Users\User\Assignment2Dir>git commit -m "changes to master barnch "
[master 9084520] changes to master barnch
 1 file changed, 1 insertion(+)
 create mode 100644 newfiledoc.doc

C:\Users\User\Assignment2Dir>|
```

11. Now merge your branch again. (Aside: you don't need to do anything to inform Git that you only want to merge things added since your previous merge. Due to the way Git works, that kind of issue simply does not come up, unlike in early versions of Subversion.)

```
C:\Users\User\Assignment2Dir>git merge branch1
Auto-merging newfiledoc.doc
CONFLICT (add/add): Merge conflict in newfiledoc.doc
Automatic merge failed; fix conflicts and then commit the result.

C:\Users\User\Assignment2Dir>|
```

12. Look at git log. Notice that there is a merge commit. Also look in gitk. Notice the DAG now shows how things forked, and then were joined up again by a merge commit.

```
C:\Users\User\Assignment2Dir>git log
commit 9084520604255dc07c469d5ffef45d030679c120 (HEAD -> master)
Author: Ronak <ronakronu02@gmail.com>
Date: Sat Jul 13 15:14:16 2024 +0530

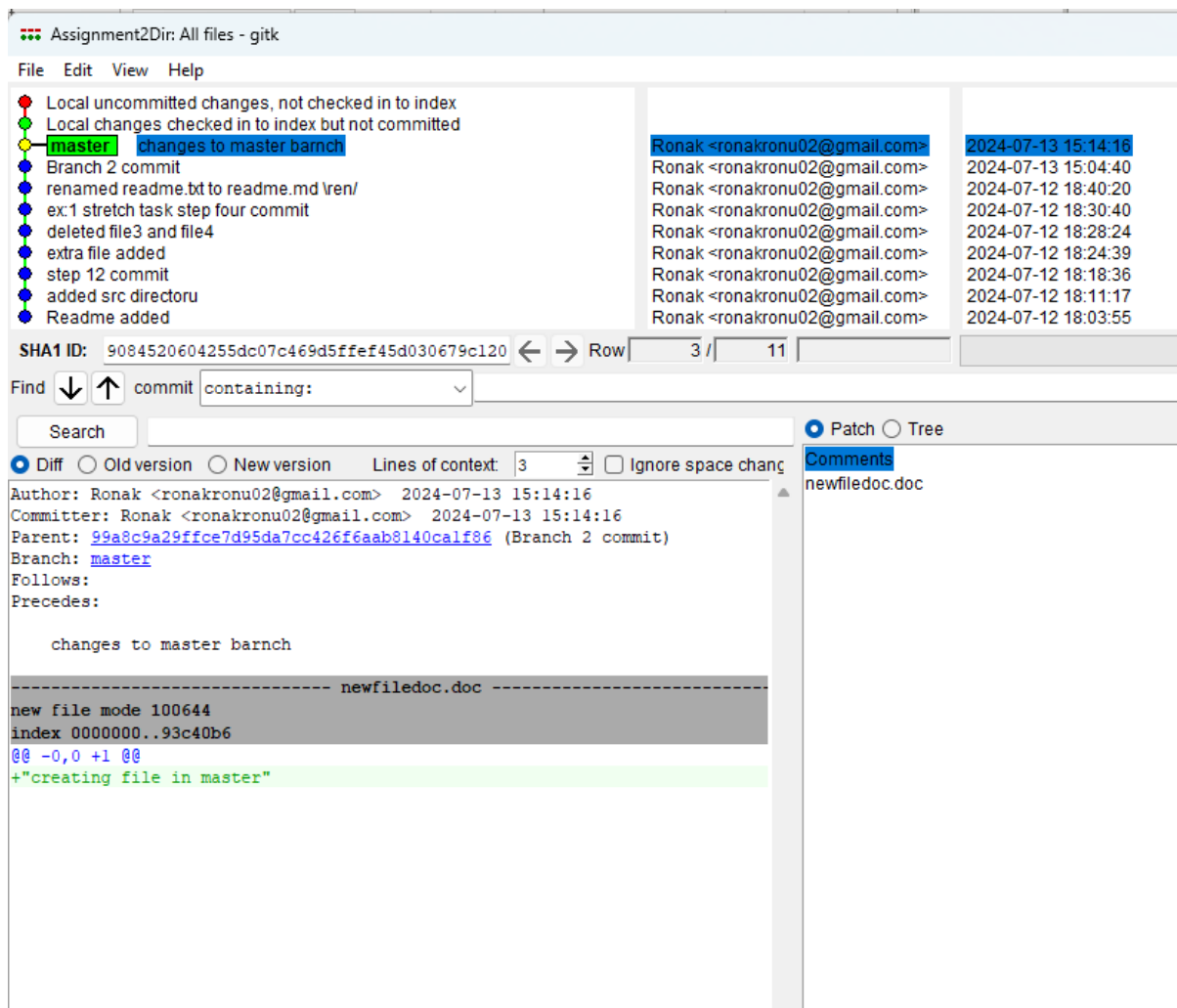
    changes to master barnch

commit 99a8c9a29ffce7d95da7cc426f6aab8140ca1f86
Author: Ronak <ronakronu02@gmail.com>
Date: Sat Jul 13 15:04:40 2024 +0530

    Branch 2 commit

commit 66d9a8af6d1b147e0dadd45b027a774a71d99063
Author: Ronak <ronakronu02@gmail.com>
Date: Fri Jul 12 18:40:20 2024 +0530

    renamed readme.txt to readme.md \ren/
```



Stretch Task

1. Once again, checkout your branch. Make a couple of commits.

```
C:\Users\User\Assignment2Dir>git commit -m "add two files in src dir"
[master b2309b0] add two files in src dir

C:\Users\User\Assignment2Dir>git log
commit b2309b06c67ca7c718b9414ba88bdb7079f7a8d1 (HEAD -> master)
Merge: 9084520 59bd778
Author: Ronak <ronakronu02@gmail.com>
Date: Sat Jul 13 15:40:43 2024 +0530

    add two files in src dir

commit 9084520604255dc07c469d5ffef45d030679c120
Author: Ronak <ronakronu02@gmail.com>
Date: Sat Jul 13 15:14:16 2024 +0530

    changes to master barnch
```

2. Return to your master branch. Make a commit there that changes the exact same line, or lines, as commits in your branch did.

```
C:\Users\User\Assignment2Dir>git commit -m "add two files in master branch"
[master 668080d] add two files in master branch
 2 files changed, 2 insertions(+)
 create mode 100644 file3.txt
 create mode 100644 file4.txt

C:\Users\User\Assignment2Dir>git commit -m "add two files in src dir"
On branch master
nothing to commit, working tree clean

C:\Users\User\Assignment2Dir>git log
commit 668080dc03e7eecf74bb06907e2c023cc9300f02 (HEAD -> master)
Author: Ronak <ronakronu02@gmail.com>
Date: Sat Jul 13 15:42:28 2024 +0530

    add two files in master branch

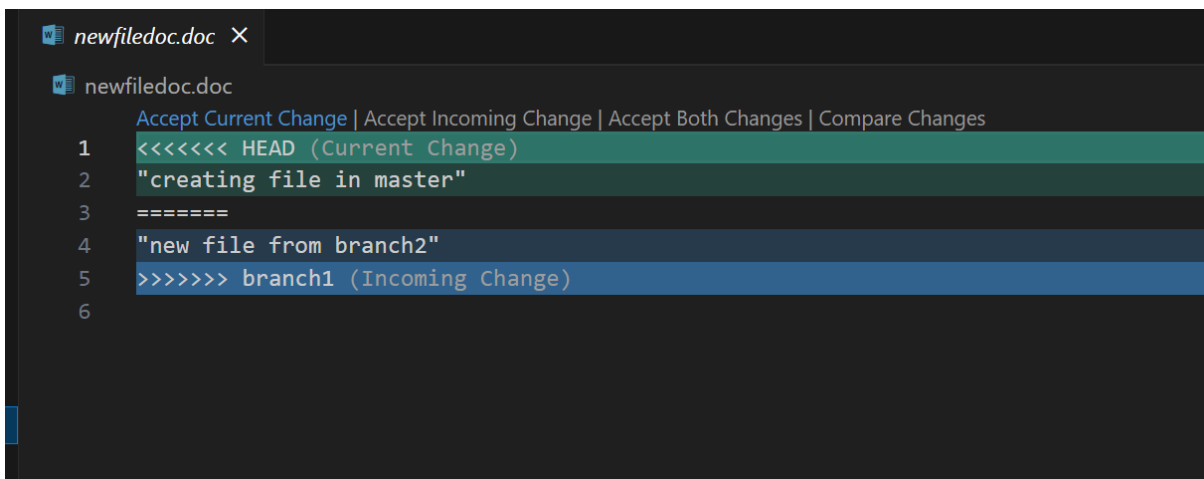
commit b2309b06c67ca7c718b9414ba88bdb7079f7a8d1
Merge: 9084520 59bd778
```

3. Now try to merge your branch. You should get a conflict.

```
C:\Users\User\Assignment2Dir>git merge branch1
Auto-merging newfiledoc.doc
CONFLICT (add/add): Merge conflict in newfiledoc.doc
Automatic merge failed; fix conflicts and then commit the result.

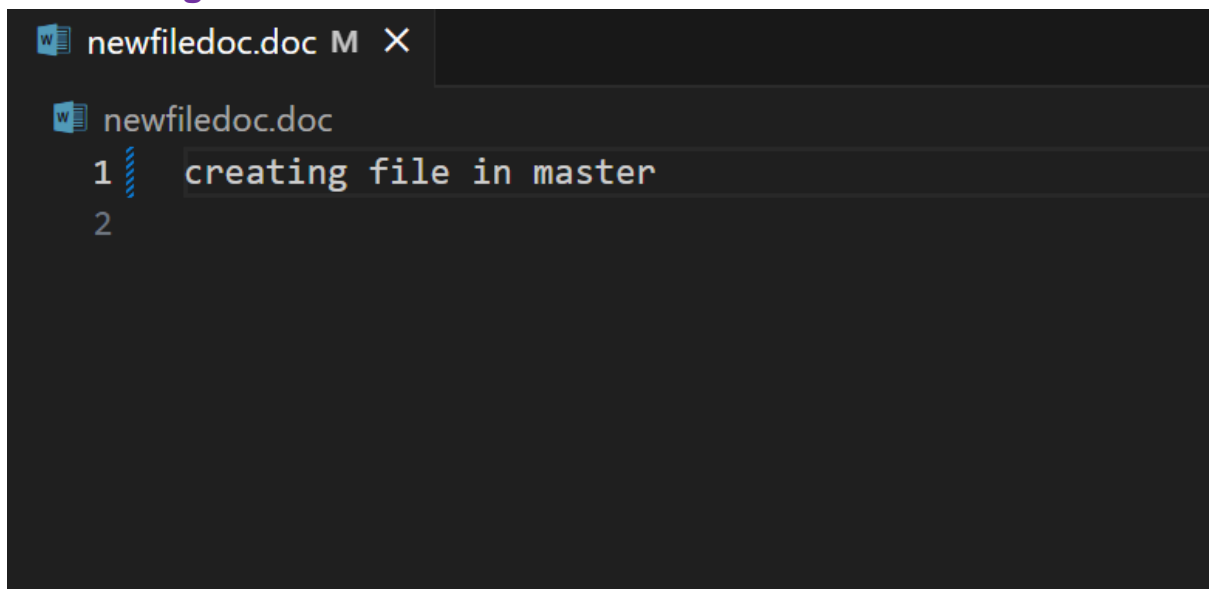
C:\Users\User\Assignment2Dir>
```

4. Open the file(s) that is in conflict. Search for the conflict marker. Edit the file to remove the conflict markers and resolve the conflict.



```
newfiledoc.doc X
newfiledoc.doc
Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes
1 <<<<<< HEAD (Current Change)
2 "creating file in master"
3 =====
4 "new file from branch2"
5 >>>>>> branch1 (Incoming Change)
6
```


New changes



5. Now try to commit. Notice that Git will not allow you to do this when you still have potentially unresolved conflicts. Look at the output of status too.

```
C:\Users\User\Assignment2Dir>git status
On branch master
Your branch is up to date with 'origin/master'.

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:   newfiledoc.doc

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

C:\Users\User\Assignment2Dir>git add newfiledoc.doc

C:\Users\User\Assignment2Dir>git commit -m "conflict resolved for newfiledoc in master"
[master 71560fc] conflict resolved for newfiledoc in master
 1 file changed, 1 insertion(+), 5 deletions(-)

C:\Users\User\Assignment2Dir>
```

6. Use the add command to add the files that you have resolved conflicts in to the staging area. Then use commit to commit the merge commit.

```

C:\Users\User\Assignment2Dir>git status
On branch master
Your branch is up to date with 'origin/master'.

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:   newfiledoc.doc

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

C:\Users\User\Assignment2Dir>git add newfiledoc.doc

C:\Users\User\Assignment2Dir>git commit -m "conflict resolved for newfiledoc in master"
[master 71560fc] conflict resolved for newfiledoc in master
1 file changed, 1 insertion(+), 5 deletions(-)

C:\Users\User\Assignment2Dir>|

```

7. Take a look at git log and gitk, and make sure things are as you expected.

```

C:\Users\User\Assignment2Dir>git log
commit 71560fc30e7263ea59a2dfbda96b02be2019a07c (HEAD -> master)
Author: Ronak <ronakronu02@gmail.com>
Date:   Tue Jul 16 09:36:41 2024 +0530

    conflict resolved for newfiledoc in master

commit 7688c2d9d9e200447ec0c726adda5c3b0a2f331b (origin/master)
Author: Ronak <ronakronu02@gmail.com>
Date:   Mon Jul 15 14:46:08 2024 +0530

    added some file

commit 48fce6b40cc63412a33af1881e39da0fb6ddebc7 (origin/branch1, branch1)
Author: Ronak Suthar <112187817+Ronak-Ronu@users.noreply.github.com>
Date:   Mon Jul 15 14:32:16 2024 +0530

    Update README.md

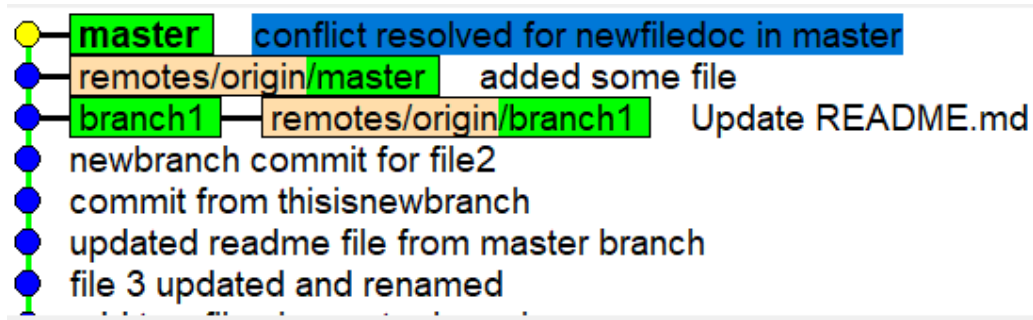
commit dd6b44a8138319d8e1e813355d748184c9ecd7c3
Author: Ronak <ronakronu02@gmail.com>
Date:   Mon Jul 15 10:04:46 2024 +0530

    newbranch commit for file2

commit 40855ba678a4212d85e6147330153c8458ae6091
Author: Ronak <ronakronu02@gmail.com>
Date:   Mon Jul 15 09:57:01 2024 +0530

    commit from thisisnewbranch

```



8. If time allows, you may wish to...

☐ Delete everything but your .git directory, then do a checkout command, to prove to yourself that this really will restore all of you current working copy.

```
C:\Users\User\Assignment2Dir>dir /a
Volume in drive C is Windows 11
Volume Serial Number is D230-3700

Directory of C:\Users\User\Assignment2Dir

16-07-2024  09.56 AM    <DIR>          .
16-07-2024  09.37 AM    <DIR>          ..
16-07-2024  09.56 AM    <DIR>          .git
               0 File(s)                0 bytes
               3 Dir(s)  112,590,827,520 bytes free
```

```

C:\Users\User\Assignment2Dir>git checkout .
Updated 15 paths from the index

C:\Users\User\Assignment2Dir>dir
Volume in drive C is Windows 11
Volume Serial Number is D230-3700

Directory of C:\Users\User\Assignment2Dir

16-07-2024  10.02 AM    <DIR>          .
16-07-2024  09.37 AM    <DIR>          ..
16-07-2024  10.02 AM                19 branch2file.txt
16-07-2024  10.02 AM                19 branch2Javafile.java
16-07-2024  10.02 AM                97 file3.txt
16-07-2024  10.02 AM                51 file4.txt
16-07-2024  10.02 AM                 5 filename.txt
16-07-2024  10.02 AM                21 newbranchfile.txt
16-07-2024  10.02 AM                26 newfilecpp.cpp
16-07-2024  10.02 AM                25 newfiledoc.doc
16-07-2024  10.02 AM                26 newfilejava.java
16-07-2024  10.02 AM                40 README.md
16-07-2024  10.02 AM                 7 somefile.txt
16-07-2024  10.02 AM    <DIR>          src
                11 File(s)                336 bytes
                 3 Dir(s) 112,586,932,224 bytes free

```

□ Create a situation where one branch has changed a file, but the other branch has deleted it. What happens when you try to merge? How will you resolve it?

```

C:\Users\User\Assignment2Dir>git branch
branch1
* master
thisisnewbranch

C:\Users\User\Assignment2Dir>git switch branch1
Switched to branch 'branch1'
Your branch is up to date with 'origin/branch1'.

```

```
C:\Users\User\Assignment2Dir>dir
Volume in drive C is Windows 11
Volume Serial Number is D230-3700
```

```
Directory of C:\Users\User\Assignment2Dir
```

```
16-07-2024  10.06 AM    <DIR>          .
16-07-2024  10.06 AM    <DIR>          ..
16-07-2024  10.02 AM                19 branch2file.txt
16-07-2024  10.02 AM                19 branch2Javafile.java
16-07-2024  10.02 AM                97 file3.txt
16-07-2024  10.02 AM                51 file4.txt
16-07-2024  10.02 AM                 5 filename.txt
16-07-2024  10.02 AM                21 newbranchfile.txt
16-07-2024  10.02 AM                26 newfilecpp.cpp
16-07-2024  10.06 AM                94 newfiledoc.doc
16-07-2024  10.02 AM                26 newfilejava.java
16-07-2024  10.02 AM                40 README.md
16-07-2024  10.02 AM    <DIR>          src
                10 File(s)                398 bytes
                3 Dir(s)  112,583,135,232 bytes free
```

```
C:\Users\User\Assignment2Dir>del file3.txt
```

```
C:\Users\User\Assignment2Dir>git add .
```

```
C:\Users\User\Assignment2Dir>git commit -m "deleted file3"
[branch1 71b7976] deleted file3
1 file changed, 3 deletions(-)
```

```
C:\Users\User\Assignment2Dir>git switch master
Switched to branch 'master'
Your branch is ahead of 'origin/master' by 1 commit.
(use "git push" to publish your local commits)
```

```
C:\Users\User\Assignment2Dir>git merge branch1
Merge made by the 'ort' strategy.
 file3.txt | 3 ---
1 file changed, 3 deletions(-)
delete mode 100644 file3.txt
```

```

C:\Users\User\Assignment2Dir>dir
Volume in drive C is Windows 11
Volume Serial Number is D230-3700

Directory of C:\Users\User\Assignment2Dir

16-07-2024  10.07 AM    <DIR>          .
16-07-2024  10.07 AM    <DIR>          ..
16-07-2024  10.02 AM                19 branch2file.txt
16-07-2024  10.02 AM                19 branch2Javafile.java
16-07-2024  10.02 AM                51 file4.txt
16-07-2024  10.02 AM                 5 filename.txt
16-07-2024  10.02 AM                21 newbranchfile.txt
16-07-2024  10.02 AM                26 newfilecpp.cpp
16-07-2024  10.07 AM                25 newfiledoc.doc
16-07-2024  10.02 AM                26 newfilejava.java
16-07-2024  10.02 AM                40 README.md
16-07-2024  10.07 AM                 7 somefile.txt
16-07-2024  10.02 AM    <DIR>          src
                        10 File(s)          239 bytes
                        3 Dir(s)  112,582,414,336 bytes free

```

☐ Look at the help page for merge, and find out how you specify a custom message for the merge commit if it is automatically generated.

```

C:\Users\User\Assignment2Dir>git branch
branch1
branchhello
branchtomerge
* master
thisisnewbranch

C:\Users\User\Assignment2Dir>git merge branchhello -m "hey merged the branchtomerge branch" --no-ff
Already up to date.

C:\Users\User\Assignment2Dir>git log
commit d95bfc5dc1bcd9d39213604168d9599a805fca1e (HEAD -> master)
Author: Ronak <ronakronu02@gmail.com>
Date: Tue Jul 16 10:20:45 2024 +0530

    hello file added

```

☐ Look at the help page for merge, and find out how to prevent Git from automatically committing the merge commit it generates, but instead give you chance to inspect it and merge it yourself.

Exercise 3

For this task, you will work in a small group. Between 2 and 4 people is about right.

Main Task

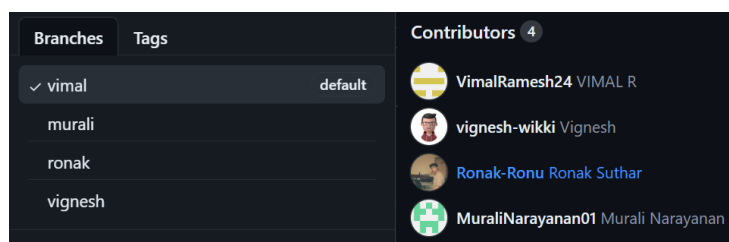
1. First, one person in the group should create a public repository using their GitHub account.

=====

2. This same person should then follow the instructions from GitHub to add a remote, and then push their repository. Do not forget the `-u` flag, as suggested by GitHub!

=====

3. All of the other members of the group should then be added as collaborators, so they can commit to the repository also.



=====

4. Next, everyone else in the group should clone the repository from GitHub. Verify that the context of the repository is what is expected.

```
C:\Users\User>git clone https://github.com/VimalRamesh24/TeamRepo.git
Cloning into 'TeamRepo'...
remote: Enumerating objects: 25, done.
remote: Counting objects: 100% (25/25), done.
remote: Compressing objects: 100% (14/14), done.
remote: Total 25 (delta 0), reused 12 (delta 0), pack-reused 0
Receiving objects: 100% (25/25), 5.57 KiB | 2.78 MiB/s, done.
```

```
C:\Users\User>cd TeamRepo
```

```
C:\Users\User\TeamRepo>git branch
```

```
* vimal
```

```
C:\Users\User\TeamRepo>git branch ronak
```

```
C:\Users\User\TeamRepo>git checkout ronak
```

```
Switched to branch 'ronak'
```

```
C:\Users\User\TeamRepo>dir
```

```
Volume in drive C is Windows 11
```

```
Volume Serial Number is D230-3700
```

```
Directory of C:\Users\User\TeamRepo
```

16-07-2024	10.22 AM	<DIR>	.
16-07-2024	10.24 AM	<DIR>	..
16-07-2024	10.22 AM		37 README.md
16-07-2024	10.22 AM		13 vignesh.md
		2 File(s)	50 bytes
		2 Dir(s)	112,577,839,104 bytes free

```
C:\Users\User\TeamRepo>git add .
```

```
C:\Users\User\TeamRepo>git commit -m "ronak updated readme"
```

```
[ronak 740dd3e] ronak updated readme'
```

```
1 file changed, 1 insertion(+)
```



```
C:\Users\User\TeamRepo>git push -u origin ronak
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 12 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 330 bytes | 330.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
remote: Create a pull request for 'ronak' on GitHub by visiting:
remote:   https://github.com/VimalRamesh24/TeamRepo/pull/new/ronak
remote:
To https://github.com/VimalRamesh24/TeamRepo.git
 * [new branch]      ronak -> ronak
branch 'ronak' set up to track 'origin/ronak'.

C:\Users\User\TeamRepo>
```

5. One of the group members who just cloned should now make a local commit, then push it.
Everyone should verify that when they pull, that commit is added to their local repository
(use git log to check for it).

```

C:\Users\User\TeamRepo>git log
commit 740dd3efe46d10d399a532eeba1a291de0c11cc8 (HEAD -> ronak, origin/ronak)
Author: Ronak <ronakronu02@gmail.com>
Date: Tue Jul 16 10:25:26 2024 +0530

    ronak updated readme'

commit a98f01b83943c860e56d97eea0b1fac9f5f270d3 (origin/vimal, origin/HEAD, vimal)
Author: VIMAL R <123486203+VimalRamesh24@users.noreply.github.com>
Date: Tue Jul 16 10:16:36 2024 +0530

    Update README.md

commit e6c63365e9090ae3664142c1339533c3367dd8e8
Author: VIMAL R <123486203+VimalRamesh24@users.noreply.github.com>
Date: Tue Jul 16 10:11:01 2024 +0530

    Update README.md

commit 475ad80536109f4fadedd59fbc7a078a36f973f0
Merge: acea6fa f60c1ce
Author: Vignesh <96247274+vignesh-wikki@users.noreply.github.com>
Date: Tue Jul 16 10:03:50 2024 +0530

    Merge pull request #1 from VimalRamesh24/vignesh

    first commit

commit f60c1ceab991011c34225981d75558cd6c7440ac
Author: vignesh-wikki <vignesha445@gmail.com>
Date: Tue Jul 16 10:02:20 2024 +0530

    first commit

commit acea6faf8c56e90263278ab6b24f43aed0d1b63a

```

6. Look at each other's git log output. Notice how the SHA-1 is the same for a given commit across every copy of the repository. Why is this important?

```

C:\Users\User\TeamRepo>git push -u origin ronak
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 12 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 330 bytes | 330.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
remote: Create a pull request for 'ronak' on GitHub by visiting:
remote:   https://github.com/VimalRamesh24/TeamRepo/pull/new/ronak
remote:
To https://github.com/VimalRamesh24/TeamRepo.git
 * [new branch]      ronak -> ronak
branch 'ronak' set up to track 'origin/ronak'.

C:\Users\User\TeamRepo>

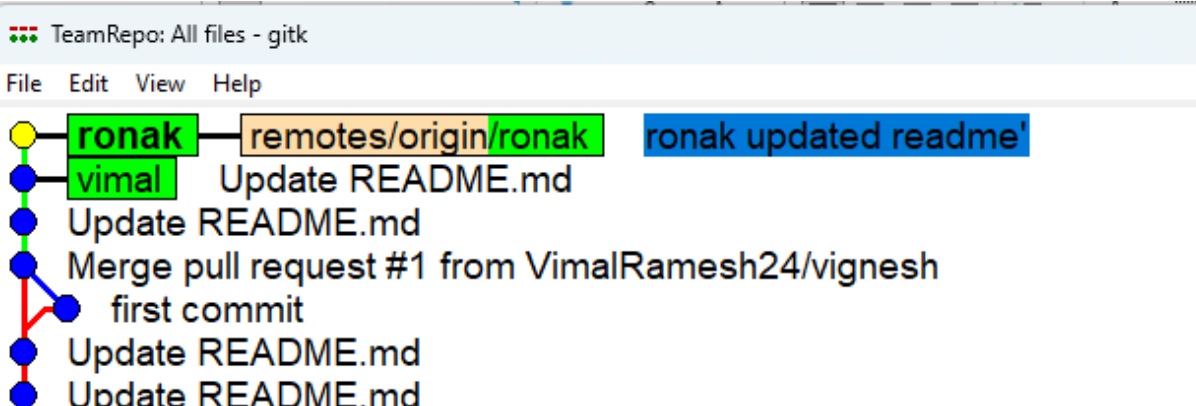
```

7. Two members of the group should now make a commit locally, and race to push it. To keep things simple, be sure to edit different files. What happens to the runner-up?

8. The runner-up should now pull. As a group, look at the output of the command.

Additionally, look at the git log, and notice that there is a merge commit. You may also wish to view the DAG in gitk.

```
C:\Users\User\TeamRepo>git pull
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (1/1), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 912 bytes | 76.00 KiB/s, done.
From https://github.com/VimalRamesh24/TeamRepo
+ a98f01b...84955a4 vimal    -> origin/vimal    (forced update)
Already up to date.
```



9. Repeat the last two steps a couple of times, to practice.

Stretch Task

1. Now create a situation where two group members both edit the same line in the same file and commit it locally. Race to push.

2. When the runner-up does a pull, they should get a merge conflict.

3. Look as a group at the file in conflict, and resolve it.

4. Use the add command to stage the fix, and then use commit to make the merge_commit.

Notice how this procedure is exactly the one you got used to when resolving conflicts in

Branches

Exercise 4

Main Task

1. Make a commit, and make a silly typo in the commit message.

```
C:\Users\User\Assignment2Dir>git status
On branch master
nothing to commit, working tree clean

C:\Users\User\Assignment2Dir>echo "file 3 is changed for exercise four" >> file3.txt

C:\Users\User\Assignment2Dir>git add file3.txt

C:\Users\User\Assignment2Dir>git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   file3.txt

C:\Users\User\Assignment2Dir>git commit -m "fle 3 updttd"
[master 8293baa] fle 3 updttd
 1 file changed, 1 insertion(+)

C:\Users\User\Assignment2Dir>
```

2. Use the --amend flag to enable you to fix the commit message.

>> git commit --amend


```

C:\Users\User\Assignment2Dir>git commit --amend
[master ec342d9] file 3 updated and renamed
Date: Mon Jul 15 09:38:45 2024 +0530
1 file changed, 1 insertion(+)

C:\Users\User\Assignment2Dir>git log
commit ec342d90910451debc4ff6c45ddef7318e1c0397 (HEAD -> master)
Author: Ronak <ronakronu02@gmail.com>
Date: Mon Jul 15 09:38:45 2024 +0530

    file 3 updated and renamed

commit 668080dc03e7eecf74bb06907e2c023cc9300f02
Author: Ronak <ronakronu02@gmail.com>
Date: Sat Jul 13 15:42:28 2024 +0530

    add two files in master branch

commit b2309b06c67ca7c718b9414ba88bdb7079f7a8d1
Merge: 9084520 59bd778

```

4. Now make a commit where you make a typo in one of the files. Once again, use --amend to magic away your problems.

5. Create a branch. Make a commit.

```

Git CMD
C:\Users\User\Assignment2Dir>git branch
branch1
* master

C:\Users\User\Assignment2Dir>git checkout -b "thisisnewbranch"
Switched to a new branch 'thisisnewbranch'

C:\Users\User\Assignment2Dir>git status
On branch thisisnewbranch
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    newbranchfile.txt

nothing added to commit but untracked files present (use "git add" to track)
C:\Users\User\Assignment2Dir>

```

```
C:\Users\User\Assignment2Dir>git status
On branch thisisnewbranch
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    newbranchfile.txt

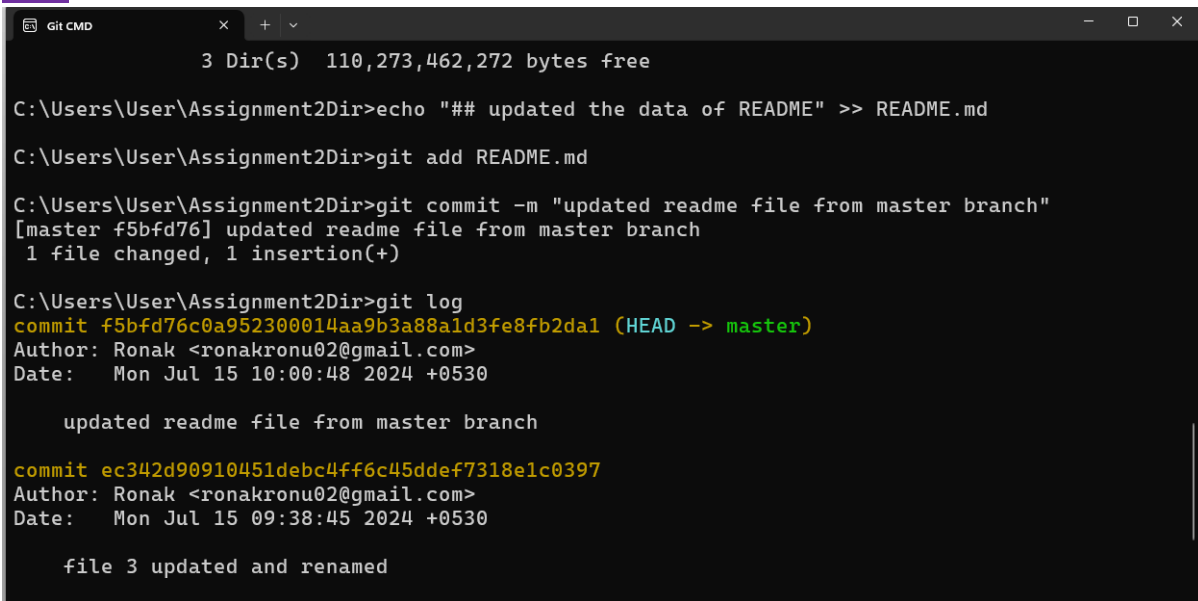
nothing added to commit but untracked files present (use "git add" to track)

C:\Users\User\Assignment2Dir>git add newbranchfile.txt

C:\Users\User\Assignment2Dir>git commit -m "commit from thisisnewbranch"
[thisisnewbranch 928ef00] commit from thisisnewbranch
 1 file changed, 1 insertion(+)
 create mode 100644 newbranchfile.txt

C:\Users\User\Assignment2Dir>
```

6. Now switch back to your master branch. Make a (non-conflicting) commit there also.



```
Git CMD
3 Dir(s) 110,273,462,272 bytes free

C:\Users\User\Assignment2Dir>echo "## updated the data of README" >> README.md

C:\Users\User\Assignment2Dir>git add README.md

C:\Users\User\Assignment2Dir>git commit -m "updated readme file from master branch"
[master f5bfd76] updated readme file from master branch
 1 file changed, 1 insertion(+)

C:\Users\User\Assignment2Dir>git log
commit f5bfd76c0a952300014aa9b3a88ald3fe8fb2da1 (HEAD -> master)
Author: Ronak <ronakronu02@gmail.com>
Date: Mon Jul 15 10:00:48 2024 +0530

    updated readme file from master branch

commit ec342d90910451debc4ff6c45ddef7318e1c0397
Author: Ronak <ronakronu02@gmail.com>
Date: Mon Jul 15 09:38:45 2024 +0530

    file 3 updated and renamed
```


7. Now switch back to your branch.

```
C:\Users\User\Assignment2Dir>git switch thisisnewbranch
Switched to branch 'thisisnewbranch'

C:\Users\User\Assignment2Dir>git log
commit 928ef00b1d687b9365c91f880127fc545086b3b7 (HEAD -> thisisnewbranch)
Author: Ronak <ronakronu02@gmail.com>
Date: Mon Jul 15 09:57:01 2024 +0530

    commit from thisisnewbranch

commit ec342d90910451debc4ff6c45ddef7318e1c0397
Author: Ronak <ronakronu02@gmail.com>
Date: Mon Jul 15 09:38:45 2024 +0530

    file 3 updated and renamed
```

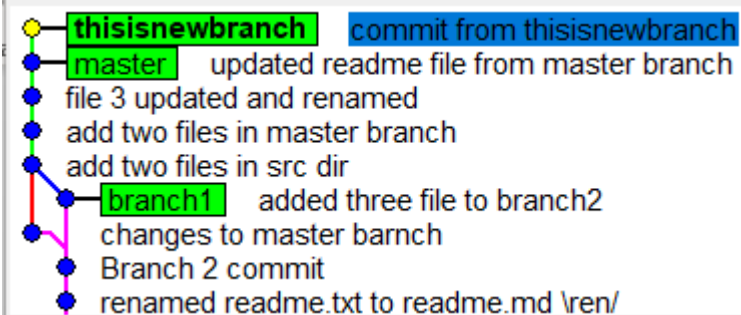
8. Use the rebase command in your branch. Look at the DAG in gitk, and note that you have the commit from the master branch, but no merge commit.

```
Git CMD

C:\Users\User\Assignment2Dir>git branch
  branch1
  master
* thisisnewbranch

C:\Users\User\Assignment2Dir>git rebase master
Successfully rebased and updated refs/heads/thisisnewbranch.

C:\Users\User\Assignment2Dir>
```



9. Make one more commit in your branch.

```
C:\Users\User\Assignment2Dir>echo "file3 change from newbranch" >> file3.txt

C:\Users\User\Assignment2Dir>git add file3.txt

C:\Users\User\Assignment2Dir>git commit -m "newbranch commit for file2"
[thisisnewbranch 2b8db8a] newbranch commit for file2
1 file changed, 1 insertion(+)

C:\Users\User\Assignment2Dir>git log
commit 2b8db8a79b5a1de53de53a20c9e7df9a79a64 (HEAD -> thisisnewbranch)
Author: Ronak <ronakronu02@gmail.com>
Date: Mon Jul 15 10:04:46 2024 +0530

    newbranch commit for file2

commit 40855ba678a4212d85e6147330153c8458ae6091
Author: Ronak <ronakronu02@gmail.com>
Date: Mon Jul 15 09:57:01 2024 +0530

    commit from thisisnewbranch
```

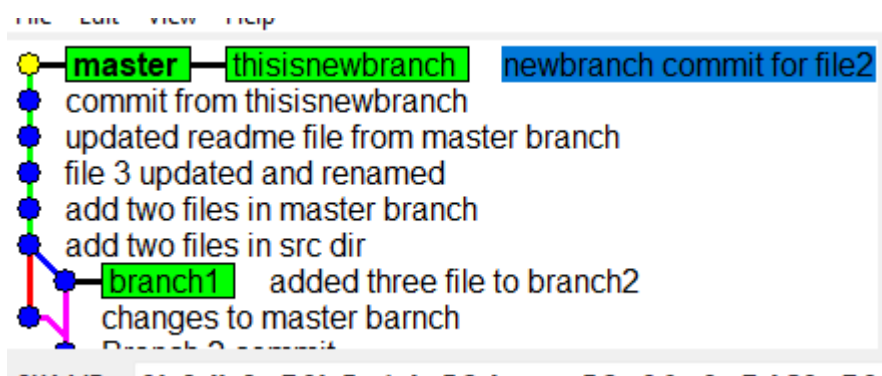
10. Return to master. Merge your branch. Notice how, thanks to the rebase, this is a fast forward merge.

```
C:\Users\User\Assignment2Dir>git branch
branch1
master
* thisisnewbranch

C:\Users\User\Assignment2Dir>git switch master
Switched to branch 'master'

C:\Users\User\Assignment2Dir>git branch
branch1
* master
thisisnewbranch

C:\Users\User\Assignment2Dir>git merge thisisnewbranch
Updating f5bfd76..2b8db8a
Fast-forward
 file3.txt          | 1 +
 newbranchfile.txt | 1 +
 2 files changed, 2 insertions(+)
 create mode 100644 newbranchfile.txt
```



Stretch Task

1. Find somebody from your team from the previous exercise. Have them push a commit to the central repository.

2. Make a commit locally yourself also. Note that you should not this point.

3. Try to push, and watch it fail.

4. Now, pull but using the --rebase flag. have pulled their commit at

5. Use git log and gitk to verify that there is no merge commit, and the DAG is linear.

6. Notice that your commit is the latest one, even though temporally the other member of your team made their commit afterwards. Why is this?

Exercise 5

Any time we have for this exercise, you are free to spend practicing whatever you find most interesting, or feel you have not fully grasped from the previous exercises and want another go through. Refer to the final section of the course for features you might like to explore.