

Git Exercises

Exercise 1

Main Task

1. Create a new directory and change into it.

MINGW64/c:/Users/atik.shaikh/Desktop/Assignments/22nd August/Demo_Git_Repo

```
Atik.Shaikh@PUN-DE-LEFT88T MINGW64 /
$ ls
LICENSE.txt      cmd/  git-bash.exe*  proc/  unins000.exe*
ReleaseNotes.html dev/  git-cmd.exe*   tmp/   unins000.msg
bin/             etc/  mingw64/       unins000.dat  usr/

Atik.Shaikh@PUN-DE-LEFT88T MINGW64 /
$ cd ..

Atik.Shaikh@PUN-DE-LEFT88T MINGW64 /
$ ls
LICENSE.txt      cmd/  git-bash.exe*  proc/  unins000.exe*
ReleaseNotes.html dev/  git-cmd.exe*   tmp/   unins000.msg
bin/             etc/  mingw64/       unins000.dat  usr/

Atik.Shaikh@PUN-DE-LEFT88T MINGW64 /
$ cd 'C:\Users\atik.shaikh\Desktop\Assignments\22nd August'

Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd August
$ mkdir Demo_Git_Repo

Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd August
$ cd Demo_Git_Repo

Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd August/Demo_Git_Re
po
$ |
```

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

Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd August/Demo_Git_Repo

\$ git init

Initialized empty Git repository in C:/Users/atik.shaikh/Desktop/Assignments/22nd August/Demo_Git_Repo/.git/

Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd August/Demo_Git_Repo (master)

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

Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd August/Demo_Git_Repo (master)

\$ git rev-parse --is-inside-work-tree

True

4. Create a README file.

Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd August/Demo_Git_Repo (master)

\$ touch Readme.txt

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

Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd August/Demo_Git_Repo (master)

\$ ls

Readme.txt

Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd August/Demo_Git_Repo (master)

\$ git status -s

?? Readme.txt

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

Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd August/Demo_Git_Repo (master)

```
$ git add Readme.txt
```

```
Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd August/Demo_Git_Repo (master)
$ git status -s
A Readme.txt
```

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

```
Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd August/Demo_Git_Repo (master)
$ git commit -m 'This is my first commit for the Readme File'
[master (root-commit) 755131b] This is my first commit for the Readme File
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 Readme.txt
```

```
Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd August/Demo_Git_Repo (master)
$ git status -s
```

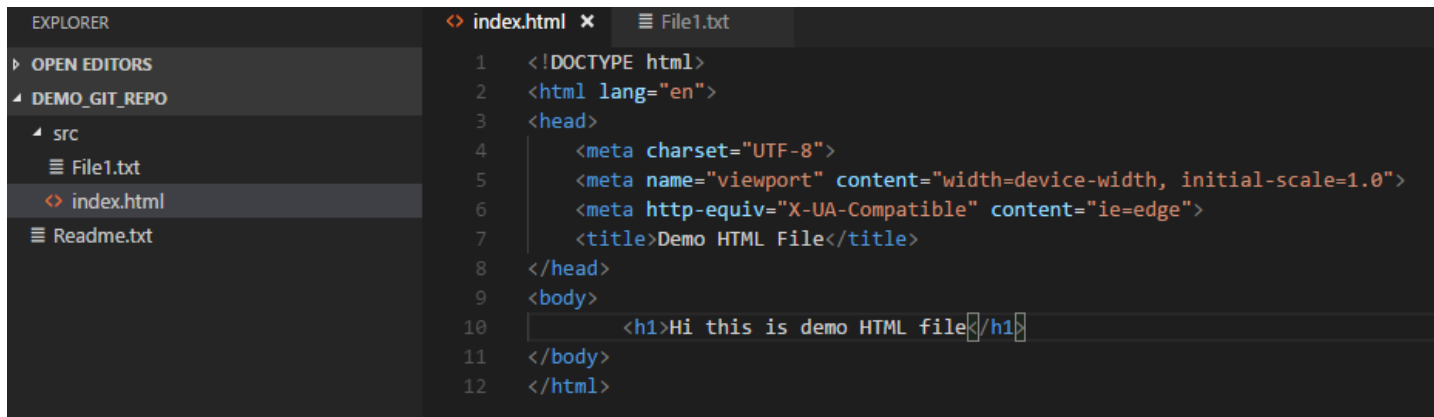
```
Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd August/Demo_Git_Repo (master)
$ git log
commit 755131b6f6d1cd8f6f8d9343e76bfead63c54f84 (HEAD -> master)
Author: Atik Shaikh <atik.shaikh@emtecinc.com>
Date: Mon Aug 22 09:39:17 2022 +0530
```

This is my first commit for the Readme File

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

```
$ mkdir src
```

```
Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd August/Demo_Git_Repo (master)
$ ls
Readme.txt src/
```



```
EXPLORER
OPEN EDITORS
DEMO_GIT_REPO
  src
    File1.txt
    index.html
    Readme.txt

index.html x  File1.txt
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <meta http-equiv="X-UA-Compatible" content="ie=edge">
7   <title>Demo HTML File</title>
8 </head>
9 <body>
10   <h1>Hi this is demo HTML file</h1>
11 </body>
12 </html>
```

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 add src/index.html
```

```
Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd August/Demo_Git_Repo (master)
$ git status -s
A src/index.html
```

```
?? src/File1.txt
```

```
Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd August/Demo_Git_Repo (master)
$ git add src/File1.txt
```

```
Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd August/Demo_Git_Repo (master)
$ git status -s
A src/File1.txt
A src/index.html
```

```
Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd August/Demo_Git_Repo (master)
$ git commit -m 'Second commit when 2 files are added in the src directory'
[master e4ce9a7] Second commit when 2 files are added in the src directory
2 files changed, 13 insertions(+)
create mode 100644 src/File1.txt
create mode 100644 src/index.html
```

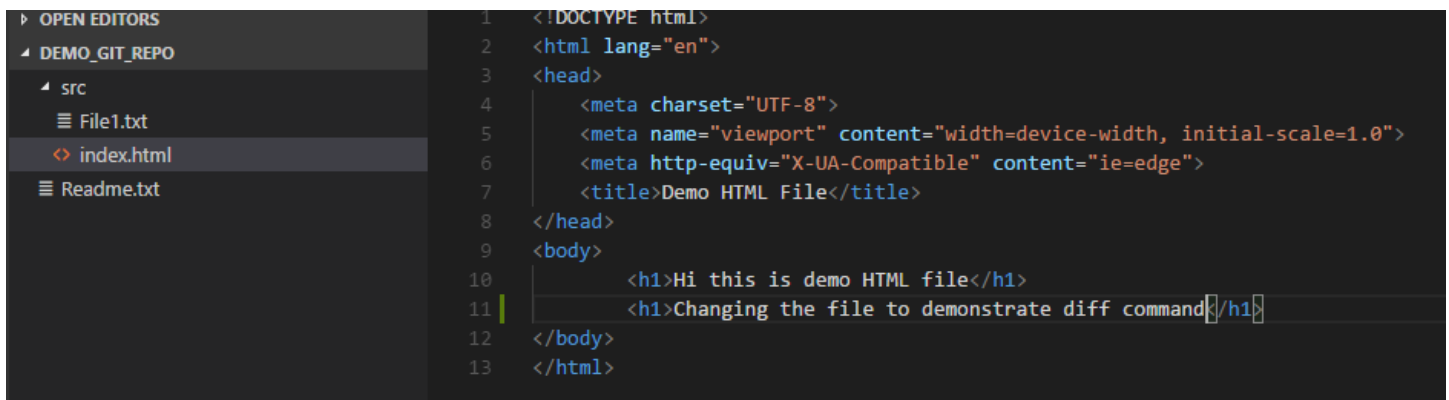
```
Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd August/Demo_Git_Repo (master)
$ git log
commit e4ce9a77dc5986f1527021fe638dec610947db05 (HEAD -> master)
Author: Atik Shaikh <atik.shaikh@emtecinc.com>
Date: Mon Aug 22 09:45:26 2022 +0530
```

Second commit when 2 files are added in the src directory

```
commit 755131b6f6d1cd8f6f8d9343e76bfead63c54f84
Author: Atik Shaikh <atik.shaikh@emtecinc.com>
Date: Mon Aug 22 09:39:17 2022 +0530
```

This is my first commit for the Readme File

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



```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <meta http-equiv="X-UA-Compatible" content="ie=edge">
7   <title>Demo HTML File</title>
8 </head>
9 <body>
10   <h1>Hi this is demo HTML file</h1>
11   <h1>Changing the file to demonstrate diff command</h1>
12 </body>
13 </html>
```

```
$ git diff
diff --git a/src/index.html b/src/index.html
index 8359b4f..8af5e0f 100644
--- a/src/index.html
+++ b/src/index.html
@@ -8,5 +8,6 @@
```

```

</head>
<body>
  <h1>Hi this is demo HTML file</h1>
+   <h1>Changing the file to demonstrate diff command</h1>
</body>
</html>
\ No newline at end of file

```

- 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.)

```

Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd August/Demo_Git_Repo (master)
$ git status -s
M src/index.html

```

```

Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd August/Demo_Git_Repo (master)
$ git add src/index.html

```

```

Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd August/Demo_Git_Repo (master)
$ git status -s
M src/index.html

```

```

Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd August/Demo_Git_Repo (master)
$ git diff

```

^In the above command there will be no output since diff cmd gives difference between the files which are staged and unstaged ones. If we want to check the difference of things in the staging area we have to use the '\$git status -s' cmd.

- 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 status -s
MM src/index.html

```

Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd August/Demo_Git_Repo (master)

\$ git diff

diff --git a/src/index.html b/src/index.html

index 8af5e0f..7ab60a6 100644

--- a/src/index.html

+++ b/src/index.html

@@ -9,5 +9,6 @@

<body>

<h1>Hi this is demo HTML file</h1>

<h1>Changing the file to demonstrate diff command</h1>

+ <h1>'I'm making change to the file which is in staging area but not yet committed</h1>

</body>

</html>

\ No newline at end of file

\$ git add .

Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd August/Demo_Git_Repo (master)

\$ git status -s

M src/index.html

Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd August/Demo_Git_Repo (master)

\$ git commit -m 'This commit is done after the demonstration of status and diff cmds'

[master 61a6858] This commit is done after the demonstration of status and diff cmds

1 file changed, 2 insertions(+)

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

Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd August/Demo_Git_Repo (master)

\$ git log

commit 61a685862eafff42ba6c7f916451c1fc1e81a9ad (HEAD -> master)

Author: Atik Shaikh <atik.shaikh@emtecinc.com>

Date: Mon Aug 22 10:03:15 2022 +0530

This commit is done after the demonstration of status and diff cmds

commit e4ce9a77dc5986f1527021fe638dec610947db05 (HEAD -> master)

Author: Atik Shaikh <atik.shaikh@emtecinc.com>

Date: Mon Aug 22 09:45:26 2022 +0530

Second commit when 2 files are added in the src directory

commit 755131b6f6d1cd8f6f8d9343e76bfead63c54f84

Author: Atik Shaikh <atik.shaikh@emtecinc.com>

Date: Mon Aug 22 09:39:17 2022 +0530

This is my first commit for the Readme File

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?

Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd August/Demo_Git_Repo (master)

\$ git checkout 61a685862eafff42ba6c7f916451c1fc1e81a9ad

Note: switching to '61a685862eafff42ba6c7f916451c1fc1e81a9ad'.

You are in 'detached HEAD' state. You can look around, make experimental changes and commit them, and you can discard any commits you make in this state without impacting any branches by switching back to a branch.

If you want to create a new branch to retain commits you create, you may do so (now or later) by using -c with the switch command. Example:

```
git switch -c <new-branch-name>
```

Or undo this operation with:

```
git switch -
```

Turn off this advice by setting config variable advice.detachedHead to false

HEAD is now at 61a6858 This commit is done after the demonstration of status and diff cmds

Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd August/Demo_Git_Repo ((61a6858...))

\$ git switch -

Switched to branch 'master'

Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd August/Demo_Git_Repo (master)

\$

^ So here we have to switch to a specific commit out of many commits using the checkout cmd and then use switch cmd again to like switch to our normal default master. Minimum characters needed are 7 for identifying the commit id.

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

Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd August/Demo_Git_Repo (master)

\$ git status -s

?? src/New_File.txt

Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd August/Demo_Git_Repo (master)

\$ git add src/New_File.txt

```
Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd August/Demo_Git_Repo (master)
$ git status -s
A src/New_File.txt
```

```
Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd August/Demo_Git_Repo (master)
$ git commit -m 'This is the commit made after adding the New_File.txt'
[master 46ae74c] This is the commit made after adding the New_File.txt
1 file changed, 1 insertion(+)
create mode 100644 src/New_File.txt
```

```
Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd August/Demo_Git_Repo (master)
$ git log
commit 46ae74c8cc7dfd58d8367cdb5506ca9bc5315d29 (HEAD -> master)
Author: Atik Shaikh <atik.shaikh@emtecinc.com>
Date: Mon Aug 22 10:13:20 2022 +0530
```

This is the commit made after adding the New_File.txt

```
commit 61a685862eafff42ba6c7f916451c1fc1e81a9ad
Author: Atik Shaikh <atik.shaikh@emtecinc.com>
Date: Mon Aug 22 10:03:15 2022 +0530
```

This commit is done after the demonstration of status and diff cmds

```
commit e4ce9a77dc5986f1527021fe638dec610947db05
Author: Atik Shaikh<atik.shaikh@emtecinc.com>
Date: Mon Aug 22 09:45:26 2022 +0530
```

Second commit when 2 files are added in the src directory

```
commit 755131b6f6d1cd8f6f8d9343e76bfead63c54f84
Author: Atik Shaikh<atik.shaikh@emtecinc.com>
Date: Mon Aug 22 09:39:17 2022 +0530
```

This is my first commit for the Readme File

Stretch Task

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

```
$ git rm src/New_File.txt
rm 'src/New_File.txt'
```

```
Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd August/Demo_Git_Repo (master)
```

```
$ git status -s
```

```
D src/New_File.txt
```

```
Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd August/Demo_Git_Repo (master)
```

```
$ git commit -m 'This commit is made after the deleting of New-File.txt to demonstrate git rm cmd'
```

```
[master ff56086] This commit is made after the deleting of New-File.txt to demonstrate git rm cmd
```

```
1 file changed, 1 deletion(-)
```

```
delete mode 100644 src/New_File.txt
```

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.

```
Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd August/Demo_Git_Repo (master)
```

```
$ git status -s
```

-Deleted via windows explorer and executed the below cmds

```
Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd August/Demo_Git_Repo (master)
```

```
$ git status -s
```

```
D src/File1.txt
```

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 status -s
```

```
D src/File1.txt
```

-Difference is that using git rm we got it staged automatically whereas in normal windows del it is unstaged.

```
$ git commit -m 'This commit is made to demonstrate difference between git rm and normal del'
```

```
[master 1509d05] This commit is made to demonstrate difference between git rm and normal del
```

```
1 file changed, 1 deletion(-)
```

```
delete mode 100644 src/File1.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.

```
$ ls
```

```
Readme.txt src/
```

```
Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd August/Demo_Git_Repo (master)
```

```
$ git mv Readme.txt Renamed_File.txt
```

```
Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd August/Demo_Git_Repo (master)
```

```
$ git status -s
```


R Readme.txt -> Renamed_File.txt

```
Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd August/Demo_Git_Repo (master)
$ git commit -m 'This commit is made after renaming the Readme.txt file'
[master 90f074f] This commit is made after renaming the Readme.txt file
1 file changed, 0 insertions(+), 0 deletions(-)
rename Readme.txt => Renamed_File.txt (100%)
```

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.) 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?

```
-Renamed using the OS and then executing the below cmds
$ git status -s
D Renamed_File.txt
?? Renamed_using_OS.txt
```

We are getting unstaged changes while doing using the normal OS . No we wont get a right outcome since it is still unstaged and we need to stage it once to commit it. We can use the git mv cmd instead of this normal OS renaming so that it wont lose the file. No, Git havnent figured out that I have renamed a file instead it is thinking I have deleted the previous file and added a new file (renamed file) but once I add this to staging area and then show the status it shows the we have renamed it actually.

```
$ git add .
```

```
Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd August/Demo_Git_Repo (master)
$ git status -s
R Renamed_File.txt -> Renamed_using_OS.txt
```

```
Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd August/Demo_Git_Repo (master)
$ git commit -m 'This commit is done to demonstrate the difference between git mv (renaming) and normal
rename using OS'
[master d55cb1e] This commit is done to demonstrate the difference between git mv (renaming) and normal
rename using OS
1 file changed, 0 insertions(+), 0 deletions(-)
rename Renamed_File.txt => Renamed_using_OS.txt (100%)
```

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

```
$ git log -n 5
commit d55cb1e37c544e280d6ad1198c167e06244f1a1f (HEAD -> master)
Author: Atik Shaikh <atik.shaikh@emtecinc.com>
Date: Mon Aug 22 10:38:45 2022 +0530
```

This commit is done to demonstrate the difference between git mv (renaming) and normal rename using OS

```
commit 90f074f7ba09d8435dc6334e3bde700f40ea423f
Author: Atik Shaikh <atik.shaikh@emtecinc.com>
```

Date: Mon Aug 22 10:28:53 2022 +0530

This commit is made after renaming the Readme.txt file

commit 1509d058dabacef6b84d6d3527726f512968343c

Author: Atik Shaikh <atik.shaikh@emtecinc.com>

Date: Mon Aug 22 10:25:49 2022 +0530

This commit is made to demonstrate difference between git rm and normal del

commit ff560863c7a1fe697164632f3c96879a4b46c3d2

Author: Atik Shaikh <atik.shaikh@emtecinc.com>

Date: Mon Aug 22 10:20:39 2022 +0530

This commit is made after the deleting of New-File.txt to demonstrate git rm cmd

commit 46ae74c8cc7dfd58d8367cdb5506ca9bc5315d29

Author: Atik Shaikh <atik.shaikh@emtecinc.com>

Date: Mon Aug 22 10:13:20 2022 +0530

This is the commit made after adding the New_File.txt

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?
\$git diff --stat

--stat[=<width>[,<name-width>[,<count>]]]

Generate a diffstat. By default, as much space as necessary will be used for the filename part, and the rest for the graph part. Maximum width defaults to terminal width, or 80 columns if not connected to a terminal, and can be overridden by <width>. The width of the filename part can be limited by giving another width <name-width> after a comma. The width of the graph part can be limited by using --stat-graph-width=<width> (affects all commands generating a stat graph) or by setting diff.statGraphWidth=<width> (does not affect git format-patch). By giving a third parameter <count>, you can limit the output to the first <count> lines, followed by ... if there are more.

These parameters can also be set individually with --stat-width=<width>, --stat-name-width=<name-width> and --stat-count=<count>.

\$ git show --pretty="" --name-only d55cb1e37c544e280d6ad1198c167e06244f1a1f

Renamed_using_OS.txt

-This will list all the files that were touched in a commit id d55cb1e37c544e280d6ad1198c167e06244f1a1f

--stat in log cmd: The --stat option displays the number of insertions and deletions to each file altered by each commit

\$ git log --stat

commit d55cb1e37c544e280d6ad1198c167e06244f1a1f (HEAD -> master)

commit d55cb1e37c544e280d6ad1198c167e06244f1a1f (HEAD -> master)

Author: Atik Shaikh <atik.shaikh@emtecinc.com>

Date: Mon Aug 22 10:38:45 2022 +0530

This commit is done to demonstrate the difference between git mv (renaming) and normal rename using OS

Renamed_File.txt => Renamed_using_OS.txt | 0

1 file changed, 0 insertions(+), 0 deletions(-)

commit 90f074f7ba09d8435dc6334e3bde700f40ea423f

Author: Atik Shaikh <atik.shaikh@emtecinc.com>

Date: Mon Aug 22 10:28:53 2022 +0530

This commit is made after renaming the Readme.txt file

Readme.txt => Renamed_File.txt | 0

1 file changed, 0 insertions(+), 0 deletions(-)

commit 1509d058dabacef6b84d6d3527726f512968343c

Author: Atik Shaikh <atik.shaikh@emtecinc.com>

Date: Mon Aug 22 10:25:49 2022 +0530

This commit is made to demonstrate difference between git rm and normal del

src/File1.txt | 1 -

1 file changed, 1 deletion(-)

commit ff560863c7a1fe697164632f3c96879a4b46c3d2

Author: Atik Shaikh <atik.shaikh@emtecinc.com>

Date: Mon Aug 22 10:20:39 2022 +0530

This commit is made after the deleting of New-File.txt to demonstrate git rm cmd

src/New_File.txt | 1 -

1 file changed, 1 deletion(-)

commit 46ae74c8cc7dfd58d8367cdb5506ca9bc5315d29

Author: Atik Shaikh <atik.shaikh@emtecinc.com>

Date: Mon Aug 22 10:13:20 2022 +0530

This is the commit made after adding the New_File.txt

src/New_File.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.

```
$ git diff d55cb1e 61a6858
diff --git a/Renamed_using_OS.txt b/Readme.txt
similarity index 100%
rename from Renamed_using_OS.txt
rename to Readme.txt
diff --git a/src/File1.txt b/src/File1.txt
new file mode 100644
index 0000000..090e6c2
--- /dev/null
+++ b/src/File1.txt
@@ -0,0 +1 @@
+Hi there this is demo file
\ No newline at end of file
```

Using the 2 commit identifiers here we are identifying the changes between the 2 commits.

Exercise 2 – Main Task

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

```
$ git status
On branch master
nothing to commit, working tree clean
```

2. Use the branch command to create a new branch.

```
$ git branch New_Branch
```

```
Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd August/Demo_Git_Repo
(master)
```

```
$ git branch
New_Branch
* master
```

3. Use the checkout command to switch to it.

```
$ git checkout New_Branch
Switched to branch 'New_Branch'
```

```
Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd August/Demo_Git_Repo
(New_Branch)
```

```
$ git branch
* New_Branch
Master
```

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

```
$ git status -s
?? src/BranchFile.txt
```

```
Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd August/Demo_Git_Repo
(New_Branch)
```

```
$ git add .
```

```
Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd August/Demo_Git_Repo
(New_Branch)
```

```
$ git commit -m 'This is commit done after adding a new file "Branched_File.txt"'
[New_Branch 0b5f8cd] This is commit done after adding a new file "Branched_File.txt"
1 file changed, 1 insertion(+)
create mode 100644 src/BranchFile.txt
```

```
$ git status -s
M src/index.html
```

```
Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd August/Demo_Git_Repo
(New_Branch)
```

```
$ git branch
* New_Branch
master
```

Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd August/Demo_Git_Repo (New_Branch)

```
$ git add .
```

Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd August/Demo_Git_Repo (New_Branch)

```
$ git commit -m 'Second commit while being on New_Branch'
[New_Branch 1992738] Second commit while being on New_Branch
1 file changed, 1 insertion(+)
```

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

```
$ git log
```

```
commit 19927380934cda2cb275360744593fc983bf0b4c (HEAD -> New_Branch)
Author: Atik Shaikh <atik.shaikh@emtecinc.com>
Date: Mon Aug 22 11:10:39 2022 +0530
```

Second commit while being on New_Branch

```
commit 0b5f8cdeb702c379c0425c8069ff7f4440507781
Author: Atik Shaikh <atik.shaikh@emtecinc.com>
Date: Mon Aug 22 11:08:54 2022 +0530
```

This is commit done after adding a new file "Branched_File.txt"

```
commit d55cb1e37c544e280d6ad1198c167e06244f1a1f (master)
Author: Atik Shaikh <atik.shaikh@emtecinc.com>
Date: Mon Aug 22 10:38:45 2022 +0530
```

This commit is done to demonstrate the difference between git mv (renaming) and normal rename using OS

```
commit 90f074f7ba09d8435dc6334e3bde700f40ea423f
Author: Atik Shaikh <atik.shaikh@emtecinc.com>
Date: Mon Aug 22 10:28:53 2022 +0530
```

This commit is made after renaming the Readme.txt file

```
commit 1509d058dabacef6b84d6d3527726f512968343c
Author: Atik Shaikh <atik.shaikh@emtecinc.com>
Date: Mon Aug 22 10:25:49 2022 +0530
```

This commit is made to demonstrate difference between git rm and normal del

```
commit ff560863c7a1fe697164632f3c96879a4b46c3d2
```

Author: Atik Shaikh <atik.shaikh@emtecinc.com>

Date: Mon Aug 22 10:20:39 2022 +0530

This commit is made after the deleting of New-File.txt to demonstrate git rm cmd

commit 46ae74c8cc7dfd58d8367cdb5506ca9bc5315d29

Author: Atik Shaikh <atik.shaikh@emtecinc.com>

Date: Mon Aug 22 10:13:20 2022 +0530

This is the commit made after adding the New_File.txt

commit 61a685862eafff42ba6c7f916451c1fc1e81a9ad

Author: Atik Shaikh <atik.shaikh@emtecinc.com>

Date: Mon Aug 22 10:03:15 2022 +0530

This commit is done after the demonstration of status and diff cmds

6. Use the checkout command to switch back to the master branch. Run log again. Notice your commits don't show up now. Check the files also – they should have their original contents.

```
$ git checkout master
```

```
Switched to branch 'master'
```

```
Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd August/Demo_Git_Repo  
(master)
```

```
$ git branch
```

```
  New_Branch
```

```
* master
```

```
$ git log
```

```
commit d55cb1e37c544e280d6ad1198c167e06244f1a1f (HEAD -> master)
```

```
Author: Atik Shaikh <atik.shaikh@emtecinc.com>
```

```
Date: Mon Aug 22 10:38:45 2022 +0530
```

This commit is done to demonstrate the difference between git mv (renaming) and normal rename using OS

commit 90f074f7ba09d8435dc6334e3bde700f40ea423f

Author: Atik Shaikh <atik.shaikh@emtecinc.com>

Date: Mon Aug 22 10:28:53 2022 +0530

This commit is made after renaming the Readme.txt file

commit 1509d058dabacef6b84d6d3527726f512968343c

Author: Atik Shaikh <atik.shaikh@emtecinc.com>

Date: Mon Aug 22 10:25:49 2022 +0530

This commit is made to demonstrate difference between git rm and normal del

commit ff560863c7a1fe697164632f3c96879a4b46c3d2
Author: Atik Shaikh <atik.shaikh@emtecinc.com>
Date: Mon Aug 22 10:20:39 2022 +0530

This commit is made after the deleting of New-File.txt to demonstrate git rm cmd

commit 46ae74c8cc7dfd58d8367cdb5506ca9bc5315d29
Author: Atik Shaikh <atik.shaikh@emtecinc.com>
Date: Mon Aug 22 10:13:20 2022 +0530

This is the commit made after adding the New_File.txt

commit 61a685862eafff42ba6c7f916451c1fc1e81a9ad
Author: Atik Shaikh <atik.shaikh@emtecinc.com>
Date: Mon Aug 22 10:03:15 2022 +0530

This commit is done after the demonstration of status and diff cmds

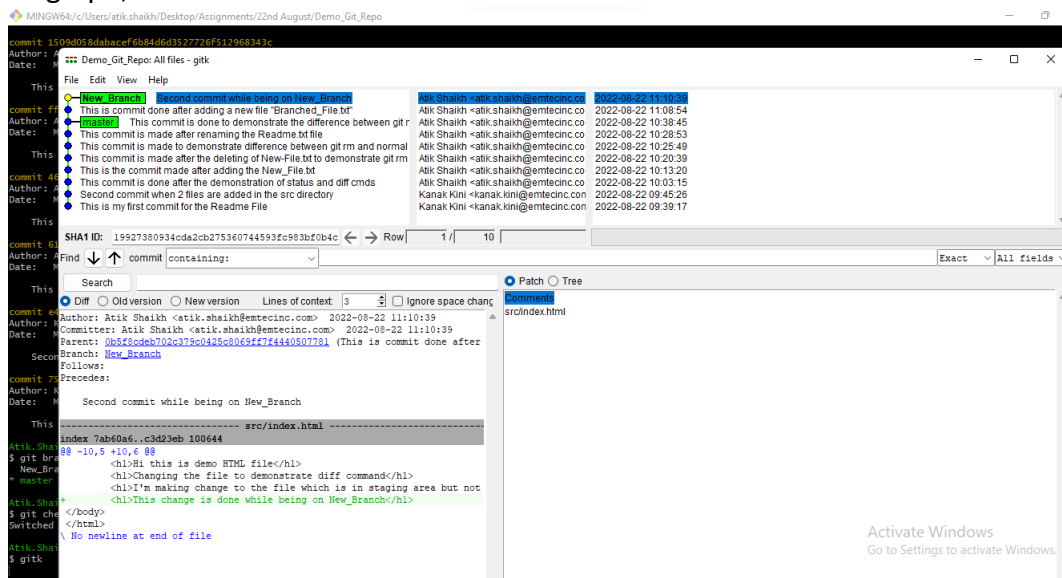
commit e4ce9a77dc5986f1527021fe638dec610947db05
Author: Atik Shaikh <atik.shaikh@emtecinc.com>
Date: Mon Aug 22 09:45:26 2022 +0530

Second commit when 2 files are added in the src directory

commit 755131b6f6d1cd8f6f8d9343e76bfead63c54f84
Author: Atik Shaikh <atik.shaikh@emtecinc.com>
Date: Mon Aug 22 09:39:17 2022 +0530

This is my first commit for the Readme File

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.

```
$ git branch
New_Branch
* master
```

```
Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd August/Demo_Git_Repo (master)
$ git checkout New_Branch
Switched to branch 'New_Branch'
```

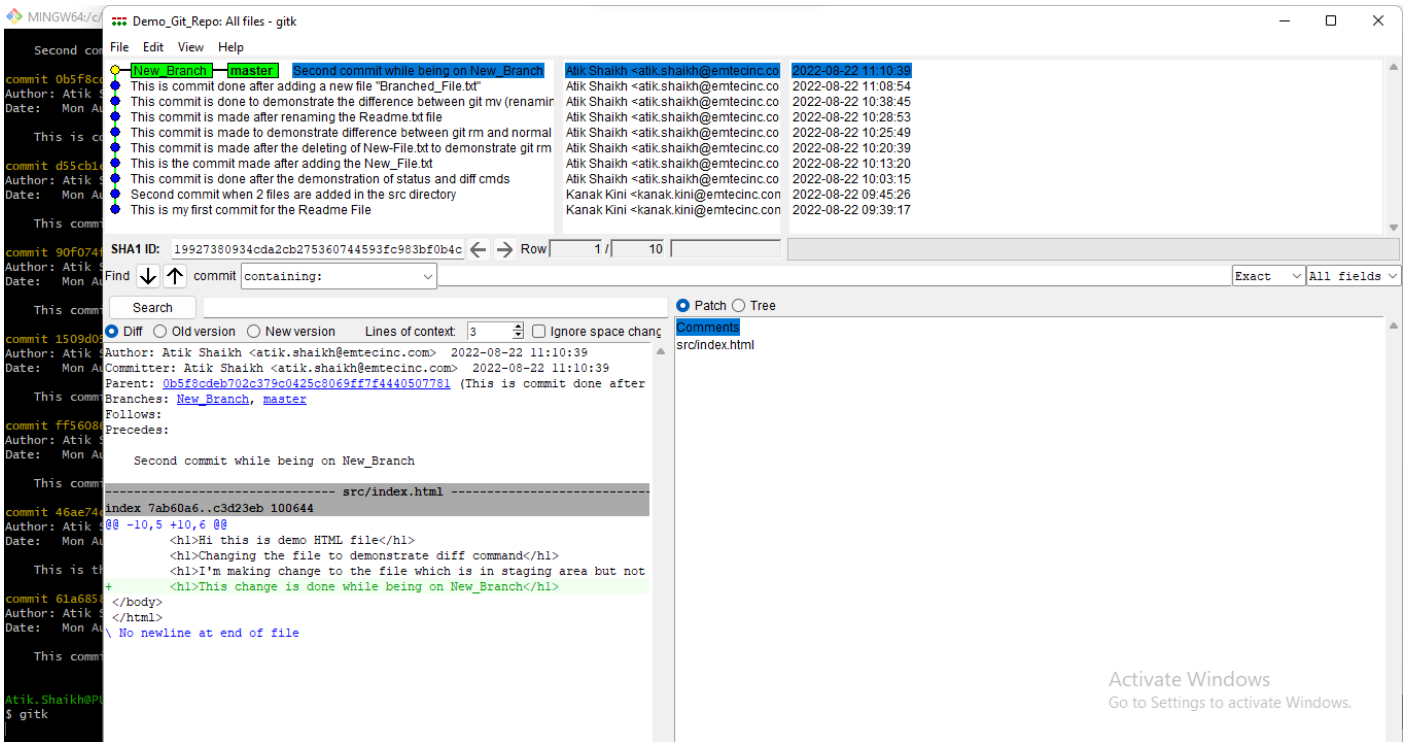
```
Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd August/Demo_Git_Repo
(New_Branch)
$ gitk
```

```
Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd August/Demo_Git_Repo
(New_Branch)
$ git checkout master
Switched to branch 'master'
```

```
Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd August/Demo_Git_Repo (master)
$ git merge New_Branch
Updating d55cb1e..1992738
Fast-forward
 src/BranchFile.txt | 1 +
 src/index.html     | 1 +
 2 files changed, 2 insertions(+)
 create mode 100644 src/BranchFile.txt
```

```
$ git log
commit 19927380934cda2cb275360744593fc983bf0b4c (HEAD -> master, New_Branch)
Author: Atik Shaikh <atik.shaikh@emtecinc.com>
Date: Mon Aug 22 11:10:39 2022 +0530
```

```
Second commit while being on New_Branch
.... ( More stats here )
```



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

Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd August/Demo_Git_Repo (master)

\$ git branch

New_Branch

* master

Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd August/Demo_Git_Repo (master)

\$ git checkout New_Branch

Switched to branch 'New_Branch'

Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd August/Demo_Git_Repo

(New_Branch)

\$ git status -s

M src/index.html

Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd August/Demo_Git_Repo

(New_Branch)

\$ git add .

Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd August/Demo_Git_Repo

(New_Branch)

\$ git commit -m 'Commit being on New_Branch after modifying index.html & BranchFile.txt'

[New_Branch 805d893] Commit being on New_Branch after modifying index.html & BranchFile.txt

1 file changed, 1 insertion(+)

Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd August/Demo_Git_Repo

(New_Branch)

\$ git status -s

M src/BranchFile.txt

```
?? src/NewFile.txt
```

```
Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd August/Demo_Git_Repo  
(New_Branch)  
$ git add .
```

```
Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd August/Demo_Git_Repo  
(New_Branch)  
$ git commit -m 'Commit done after creating NewFile.txt'  
[New_Branch 2eabff8] Commit done after creating NewFile.txt  
2 files changed, 2 insertions(+)  
create mode 100644 src/NewFile.txt
```

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.

```
$ git branch  
* New_Branch  
master
```

```
Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd August/Demo_Git_Repo  
(New_Branch)  
$ git checkout master  
Switched to branch 'master'
```

```
Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd August/Demo_Git_Repo (master)  
$ git status -s  
?? src/File_Master.html
```

```
Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd August/Demo_Git_Repo (master)  
$ git add .
```

```
Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd August/Demo_Git_Repo (master)  
$ git commit -m 'Commit done after adding File_Master.html in the master branch'  
[master 93aaa34] Commit done after adding File_Master.html in the master branch  
1 file changed, 12 insertions(+)  
create mode 100644 src/File_Master.html
```

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.)

```
$ git branch  
New_Branch  
* master
```

```
Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd August/Demo_Git_Repo (master)  
$ git merge New_Branch  
Merge made by the 'ort' strategy.  
src/BranchFile.txt | 1 +
```

```
src/NewFile.txt | 1 +
src/index.html | 1 +
3 files changed, 3 insertions(+)
create mode 100644 src/NewFile.txt
```

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.

```
$ git log
commit 7188b5e80841857e505798d111ee52af6f77be60 (HEAD -> master)
Merge: 93aaa34 2eabff8
Author: Atik Shaikh <atik.shaikh@emtecinc.com>
Date: Mon Aug 22 11:34:37 2022 +0530
```

```
Merge branch 'New_Branch'
:wq
```

```
commit 93aaa346453194d9fd5db1e5decaf13b86d9550d
Author: Atik Shaikh <atik.shaikh@emtecinc.com>
Date: Mon Aug 22 11:33:42 2022 +0530
```

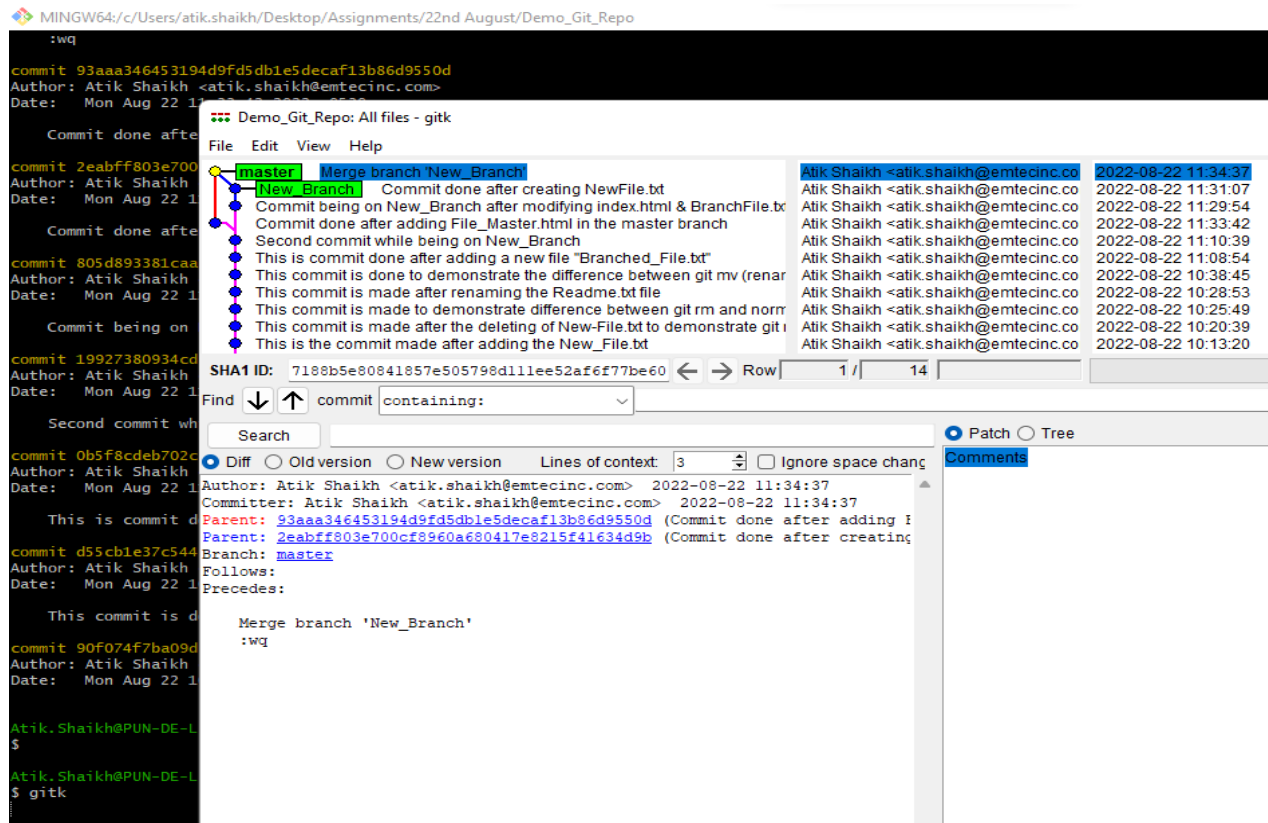
Commit done after adding File_Master.html in the master branch

```
commit 2eabff803e700cf8960a680417e8215f41634d9b (New_Branch)
Author: Atik Shaikh <atik.shaikh@emtecinc.com>
Date: Mon Aug 22 11:31:07 2022 +0530
```

Commit done after creating NewFile.txt

```
commit 805d893381caae28f034d3f75218771db4ddec1f
Author: Atik Shaikh <atik.shaikh@emtecinc.com>
Date: Mon Aug 22 11:29:54 2022 +0530
```

Commit being on New_Branch after modifying index.html & BranchFile.txt



Stretch Task:

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

```

$ git branch
New_Branch
* master

```

Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd August/Demo_Git_Repo (master)

```

$ git checkout New_Branch
Switched to branch 'New_Branch'

```

Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd August/Demo_Git_Repo (New_Branch)

```

$ git status -s
M src/BranchFile.txt
M src/index.html

```

Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd August/Demo_Git_Repo (New_Branch)

```

$ git add .

```

Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd August/Demo_Git_Repo (New_Branch)

```

$ git status -s
M src/BranchFile.txt
M src/index.html

```

Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd August/Demo_Git_Repo (New_Branch)

```
$ git commit -m 'Stretch Task 1st commit'
[New_Branch c169bb7] Stretch Task 1st commit
2 files changed, 3 insertions(+), 1 deletion(-)
```

```
Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd August/Demo_Git_Repo (New_Branch)
$ git status -s
?? src/StretchTask.txt
```

```
Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd August/Demo_Git_Repo (New_Branch)
$ git add .
```

```
Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd August/Demo_Git_Repo (New_Branch)
$ git commit -m 'Stretch Task 2nd commit'
[New_Branch 0efc949] Stretch Task 2nd commit
1 file changed, 1 insertion(+)
create mode 100644 src/StretchTask.txt
```

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

```
-Changed some lines in index.html ( Same lines which I changed while being on NewBranch)
$ git checkout master
Switched to branch 'master'
```

```
Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd August/Demo_Git_Repo (master)
$ git status -s
M src/index.html
```

```
Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd August/Demo_Git_Repo (master)
$ git add .
```

```
Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd August/Demo_Git_Repo (master)
$ git commit -m 'Commit done while being on Master Branch to demonstrate the conflict while merging'
[master fff061b] Commit done while being on Master Branch to demonstrate the conflict while merging
1 file changed, 1 insertion(+), 1 deletion(-)
```

3. Now try to merge your branch. You should get a conflict.
\$ git branch
New_Branch
* master

```
Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd August/Demo_Git_Repo (master)
$ git merge New_Branch
Auto-merging src/index.html
CONFLICT (content): Merge conflict in src/index.html
Automatic merge failed; fix conflicts and then commit the result.
```

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.

-Resolved the conflict by accepting both the incoming changes.

```
$ git status -s
```

```
M src/BranchFile.txt
```

```
A src/StretchTask.txt
```

```
UU src/index.html
```

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.

```
Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd August/Demo_Git_Repo
```

```
(master|MERGING)
```

```
$ git commit -m 'Trying to demonstrate error'
```

```
error: Committing is not possible because you have unmerged files.
```

```
hint: Fix them up in the work tree, and then use 'git add/rm <file>'
```

```
hint: as appropriate to mark resolution and make a commit.
```

```
fatal: Exiting because of an unresolved conflict.
```

```
U    src/index.html
```

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.

```
$ git add .
```

```
Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd August/Demo_Git_Repo
```

```
(master|MERGING)
```

```
$ git status -s
```

```
M src/BranchFile.txt
```

```
A src/StretchTask.txt
```

```
M src/index.html
```

```
Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd August/Demo_Git_Repo
```

```
(master|MERGING)
```

```
$ git commit -m 'Commit done after resolving the conflict while merging New_Branch with Master branch'
```

```
[master fd8ff30] Commit done after resolving the conflict while merging New_Branch with Master branch
```

```
Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd August/Demo_Git_Repo (master)
```

```
$ git status -s
```

```
Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd August/Demo_Git_Repo (master)
```

```
$ git status
```

```
On branch master
```

```
nothing to commit, working tree clean
```

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

```
$ git log
```

```
commit fd8ff3038372d9a5d49b35cc9a76d48ebed1c160 (HEAD -> master)
```

```
Merge: fff061b 0efc949
```

```
Author: Atik Shaikh <atik.shaikh@emtecinc.com>
```

```
Date: Mon Aug 22 11:51:39 2022 +0530
```

Commit done after resolving the conflict while merging New_Branch with Master branch

```
commit fff061b10df78eefae70ca23007a41350c21fa47
```

```
Author: Atik Shaikh <atik.shaikh@emtecinc.com>
```

```
Date: Mon Aug 22 11:46:36 2022 +0530
```

Commit done while being on Master Branch to demonstrate the conflict while merging

```
commit 0efc94993c231a0ad13e2b2cf5002302e2659220 (New_Branch)
```

```
Author: Atik Shaikh <atik.shaikh@emtecinc.com>
```

```
Date: Mon Aug 22 11:41:40 2022 +0530
```

Stretch Task 2nd commit

```
commit c169bb75c71ce5663728d51b9c6d4a819fb34ea9
```

```
Author: Atik Shaikh <atik.shaikh@emtecinc.com>
```

```
Date: Mon Aug 22 11:40:46 2022 +0530
```

Stretch Task 1st commit

```
commit 7188b5e80841857e505798d111ee52af6f77be60
```

```
Merge: 93aaa34 2eabff8
```

```
Author: Atik Shaikh <atik.shaikh@emtecinc.com>
```

```
Date: Mon Aug 22 11:34:37 2022 +0530
```

Merge branch 'New_Branch'

```
:wq
```

```
commit 93aaa346453194d9fd5db1e5decaf13b86d9550d
```

```
Author: Atik Shaikh <atik.shaikh@emtecinc.com>
```

```
Date: Mon Aug 22 11:33:42 2022 +0530
```

Commit done after adding File_Master.html in the master branch

```
commit 2eabff803e700cf8960a680417e8215f41634d9b
```

```
Author: Atik Shaikh <atik.shaikh@emtecinc.com>
```

```
Date: Mon Aug 22 11:31:07 2022 +0530
```

Commit done after creating NewFile.txt

```
commit 805d893381caae28f034d3f75218771db4ddec1f
```

```
Author: Atik Shaikh <atik.shaikh@emtecinc.com>
```


Date: Mon Aug 22 11:29:54 2022 +0530

Gitk

The screenshot displays the Gitk graphical user interface. The top window shows a commit history list with columns for commit hash, message, author, and date. The bottom window shows a diff view for the file `src/index.html`, comparing the current commit with its parent. The diff shows changes in HTML tags and text.

Commit History:

Commit Hash	Message	Author	Date
commit fff0611	Stretch Task 2nd commit	Atik Shaikh <atik.shaikh@emtecinc.co>	2022-08-22 11:41:40
commit 0efc94b	Stretch Task 1st commit	Atik Shaikh <atik.shaikh@emtecinc.co>	2022-08-22 11:40:46
commit c169bb7	Commit done while being on Master Branch to demonstrate the conflict v	Atik Shaikh <atik.shaikh@emtecinc.co>	2022-08-22 11:46:36
commit 7188b5d	Merge branch 'New_Branch'	Atik Shaikh <atik.shaikh@emtecinc.co>	2022-08-22 11:34:37
commit 93aaa34	Commit done after creating NewFile.txt	Atik Shaikh <atik.shaikh@emtecinc.co>	2022-08-22 11:31:07
commit 2eabff5	Commit being on New_Branch after modifying index.html & BranchFile.txt	Atik Shaikh <atik.shaikh@emtecinc.co>	2022-08-22 11:29:54
commit 805d891	Commit done after adding File_Master.html in the master branch	Atik Shaikh <atik.shaikh@emtecinc.co>	2022-08-22 11:33:42
commit 11b10d7	Second commit while being on New_Branch	Atik Shaikh <atik.shaikh@emtecinc.co>	2022-08-22 11:10:39
commit 061b10d	This is commit done after adding a new file "Branched_File.txt"	Atik Shaikh <atik.shaikh@emtecinc.co>	2022-08-22 11:08:54
commit 0efc94b	This commit is done to demonstrate the difference between git mv (renam	Atik Shaikh <atik.shaikh@emtecinc.co>	2022-08-22 10:38:45
commit 11b10d7	This commit is made after renaming the Readme.txt file	Atik Shaikh <atik.shaikh@emtecinc.co>	2022-08-22 10:28:53
commit 061b10d	This commit is made to demonstrate difference between git rm and norm	Atik Shaikh <atik.shaikh@emtecinc.co>	2022-08-22 10:25:49
commit 11b10d7	This commit is made after the deleting of New_File.txt to demonstrate git	Atik Shaikh <atik.shaikh@emtecinc.co>	2022-08-22 10:20:39
commit 061b10d	This is the commit made after adding the New_File.txt	Atik Shaikh <atik.shaikh@emtecinc.co>	2022-08-22 10:13:20
commit 11b10d7	This commit is done after the demonstration of status and diff cmds	Atik Shaikh <atik.shaikh@emtecinc.co>	2022-08-22 10:03:15
commit 061b10d	Second commit when 2 files are added in the src directory	Kanak Kini <kanak.kini@emtecinc.co>	2022-08-22 09:45:26
commit 11b10d7	This is my first commit for the Readme File	Kanak Kini <kanak.kini@emtecinc.co>	2022-08-22 09:39:17

Diff View:

Author: Atik Shaikh <atik.shaikh@emtecinc.co> 2022-08-22 11:51:39
Committer: Atik Shaikh <atik.shaikh@emtecinc.co> 2022-08-22 11:51:39
Parent: fff06110d778ee70ca23007a41350c21fa47 (Commit done while being on
Parent: 0efc94993c231a0ad13e2b2cf5002302e2659220 (Stretch Task 2nd commit)
Branch: master
Follows:
Precedes:
Commit done after resolving the conflict while merging New_Branch with 1

Diff:

```
index f860e0f,ac41c43..69276bf
@@ -11,6 -11,7 +11,8 @@
<h1>Changing the file to demonstrate diff command</h1>
<h1>I'm making change to the file which is in staging area but not
<h1>This change is done while being on New_Branch</h1>
+<h1>Changing this line while being on the master branch</h1>
+<h1>Making this change for the 1st commit after the merging task v
+<h2>Doing some more commits for the Stretch Task</h2>
```

8. Optional but will be done

Exercise 3 Main Task

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

Suraj1808 (Github Username) has created a public repo for our team of 2 (other member – Emtecatikshaikh)

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!

\$ git remote add origin https://github.com/Suraj1808/Git_Exercise3.git

Suraj.Dalvi@PUN-DE-DL6VD02 MINGW64 ~/desktop/exercises/src (master)

\$ git config -l

diff.astextplain.textconv=astextplain

filter.lfs.clean=git-lfs clean -- %f

filter.lfs.smudge=git-lfs smudge -- %f

filter.lfs.process=git-lfs filter-process

filter.lfs.required=true

http.sslbackend=openssl

http.sslcainfo=C:/Users/suraj.dalvi/AppData/Local/Programs/Git/mingw64/ssl/certs/ca-bundle.crt

core.autocrlf=true

core.fscache=true

core.symlinks=false

pull.rebase=false

credential.helper=manager-core

credential.https://dev.azure.com.usehttppath=true

init.defaultbranch=master

user.name=Suraj.Dalvi

user.email=Suraj.Dalvi@emtecinc.com

core.repositoryformatversion=0

core.filemode=false

core.bare=false

core.logallrefupdates=true

core.symlinks=false

core.ignorecase=true

remote.origin.url=https://github.com/Suraj1808/Git_Exercise3.git

remote.origin.fetch=+refs/heads/*:refs/remotes/origin/*

Suraj.Dalvi@PUN-DE-DL6VD02 MINGW64 ~/desktop/exercises/src (master)

\$ git push -u origin master

Enumerating objects: 60, done.

Counting objects: 100% (60/60), done.

Delta compression using up to 4 threads

Compressing objects: 100% (53/53), done.

Writing objects: 100% (60/60), 5.28 KiB | 385.00 KiB/s, done.

Total 60 (delta 16), reused 0 (delta 0), pack-reused 0

```
remote: Resolving deltas: 100% (16/16), done.  
To https://github.com/Suraj1808/Git_Exercise3.git  
* [new branch]    master -> master  
branch 'master' set up to track 'origin/master'.
```

3. All of the other members of the group should then be added as collaborators, so they can commit to the repository also.
Suraj added me as a collaborator in this public repo. (Done)
4. Next, everyone else in the group should clone the repository from GitHub. Verify that the context of the repository is what is expected.

```
$ git clone https://github.com/Suraj1808/Git\_Exercise3.git
```

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).

```
$ git clone https://github.com/Suraj1808/Git_Exercise3.git  
Cloning into 'Git_Exercise3'...  
remote: Enumerating objects: 60, done.  
remote: Counting objects: 100% (60/60), done.  
remote: Compressing objects: 100% (37/37), done.  
remote: Total 60 (delta 16), reused 60 (delta 16), pack-reused 0  
Receiving objects: 100% (60/60), 5.28 KiB | 771.00 KiB/s, done.  
Resolving deltas: 100% (16/16), done.
```

```
Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd  
August/Suraj_Github_Repo/Git_Exercise3 (main)  
$ ls  
Git_Exercise3/
```

```
Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd  
August/Suraj_Github_Repo/Git_Exercise3 (main)  
$ cd Git_Exercise3
```

```
Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd  
August/Suraj_Github_Repo/Git_Exercise3/Git_Exercise3 (master)  
$ ls  
readme.txt src/
```

```
Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd  
August/Suraj_Github_Repo/Git_Exercise3/Git_Exercise3 (master)  
$ git status -s  
M readme.txt
```

```
M src/exe.html
M src/exe.txt
D src/file.txt
M src/index.html
?? src/file.html
```

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?

```
Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd
August/Suraj_Github_Repo/Git_Exercise3/Git_Exercise3 (master)
$ git add .
```

```
Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd
August/Suraj_Github_Repo/Git_Exercise3/Git_Exercise3 (master)
$ git commit -m 'making changes in surajs github repo'
[master a5a3188] making changes in surajs github repo
5 files changed, 7 insertions(+), 3 deletions(-)
rename src/{file.txt => file.html} (80%)
```

```
Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd
August/Suraj_Github_Repo/Git_Exercise3/Git_Exercise3 (master)
$ git status -s
```

```
$git log
commit a5a3188ee885892eae48567ff7bfb53ad512f228 (HEAD -> master)
Author: Atik Shaikh <atik.shaikh@emtecinc.com>
Date: Mon Aug 22 13:17:37 2022 +0530
```

```
    making changes in surajs github repo
#This commit is made by me in Surajs Repo
```

```
commit f15fb6afd8193d5281507cff33a9073b920889d4 (origin/master, origin/HEAD)
Merge: ca8fe8a 13200ab
Author: Suraj.Dalvi <Suraj.Dalvi@emtecinc.com>
Date: Sun Aug 21 22:54:23 2022 -0700
```

```
    commmit after conflict
( below are the suraj's commits made to his local repo )
```

```
Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd
August/Suraj_Github_Repo/Git_Exercise3/Git_Exercise3 (master)
$ git push
Enumerating objects: 14, done.
Counting objects: 100% (14/14), done.
```

Delta compression using up to 8 threads
Compressing objects: 100% (7/7), done.
Writing objects: 100% (8/8), 810 bytes | 405.00 KiB/s, done.
Total 8 (delta 3), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (3/3), completed with 2 local objects.
To https://github.com/Suraj1808/Git_Exercise3.git
f15fb6a..a5a3188 master -> master

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?

Me being runner-up:

```
$ git status -s  
?? src/Atik.html
```

```
Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd  
August/Suraj_Github_Repo/Git_Exercise3/Git_Exercise3 (master)  
$ git add .
```

```
Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd  
August/Suraj_Github_Repo/Git_Exercise3/Git_Exercise3 (master)  
$ git commit -m 'Commit done while being a runner-up'  
[master 8a840b4] Commit done while being a runner-up  
1 file changed, 13 insertions(+)  
create mode 100644 src/Atik.html
```

```
$ git push  
To https://github.com/Suraj1808/Git\_Exercise3.git  
! [rejected] master -> master (fetch first)  
error: failed to push some refs to 'https://github.com/Suraj1808/Git_Exercise3.git'  
hint: Updates were rejected because the remote contains work that you do  
hint: not have locally. This is usually caused by another repository pushing  
hint: to the same ref. You may want to first integrate the remote changes  
hint: (e.g., 'git pull ...') before pushing again.  
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```

So the reason for ERROR is that first I have to pull the changes which are done by Suraj and then push my changes to the repo.

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.

```
$ git pull
```

```
remote: Enumerating objects: 12, done.
remote: Counting objects: 100% (12/12), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 7 (delta 3), reused 7 (delta 3), pack-reused 0
Unpacking objects: 100% (7/7), 768 bytes | 2.00 KiB/s, done.
From https://github.com/Suraj1808/Git_Exercise3
   a5a3188..adbe973  master    -> origin/master
Merge made by the 'ort' strategy.
 src/exe.html | 1 +
 src/exe.txt  | 3 ++-
 src/index.html | 1 +
 src/suraj.html | 12 ++++++++
4 files changed, 16 insertions(+), 1 deletion(-)
create mode 100644 src/suraj.html
```

Ok, so after pulling Suraj's changes my file which I had created before the pull req 'Atik.html' is not showing in the status-s even though I haven't added it in the staging area yet and then committed it.

Now, I'm pushing my changes after pulling surajs changes first.

```
$ git push
```

```
Enumerating objects: 11, done.
Counting objects: 100% (11/11), done.
Delta compression using up to 8 threads
Compressing objects: 100% (7/7), done.
Writing objects: 100% (7/7), 977 bytes | 488.00 KiB/s, done.
Total 7 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/Suraj1808/Git_Exercise3.git
   adbe973..340a4fe  master -> master
```

This log obtained in the Suraj's machine (Same is shown in mine which means it has successfully merged)

```
$ git log
```

```
commit 340a4fe7534f96023a37f289cd10890eae2c86df
Merge: 8a840b4 adbe973
Author: Atik Shaikh <atik.shaikh@emtecinc.com>
Date: Mon Aug 22 13:35:40 2022 +0530
```

```

MINGW64 Git_Exercise3: All files - gitk
commit f15f...
Merge: ca8f...
Author: Suraj
Date: Sun

commit
Suraj's commit done
Merge branch 'master' of https://github.com/Suraj1808/Git_Exercise3
this is commit for 7th task
Commit done while being a runner-up
making changes in surajs github repo
commit after conflict
commit in b1
b1 commit
commit in master
commit in master after merge
3rd commit in branching
2nd commit in branching
1st commit in branching
commit after change
commit after remove cmd
extra commit
commit after 2 changes at staged and unstaged

Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~
$ git add .
$ git commit -m 'Committing being a winner in the race to push first to the Github Repo'
$ git status
?? Unknown_File.py

Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~
$ git push
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 8 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 447 bytes | 447.00 KiB/s, done.
Total 4 (delta 2), reused 0 (delta 0), 1 file changed, 1 insertion(+)
remote: Resolving deltas: 100% (4/4), done.
To https://github.com:Suraj1808/Git_Exercise3
   master: master -> f15f... (Suraj's commit done)

Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~
$ git status
?? Unknown_File.py

```

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

Now me being a winner in the race to push first to the Github repo:

Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd August/Suraj_Github_Repo/Git_Exercise3/Git_Exercise3/src (master)

\$ git status -s

?? Unknown_File.py

Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd August/Suraj_Github_Repo/Git_Exercise3/Git_Exercise3/src (master)

\$ git add .

Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd August/Suraj_Github_Repo/Git_Exercise3/Git_Exercise3/src (master)

\$ git commit -m 'Committing being a winner in the race to push first to the Github Repo'
[master aceac48] Committing being a winner in the race to push first to the Github Repo
1 file changed, 1 insertion(+)
create mode 100644 src/Unknown_File.py

Pushing first:

\$ git push

Enumerating objects: 6, done.

Counting objects: 100% (6/6), done.

Delta compression using up to 8 threads

Compressing objects: 100% (4/4), done.

Writing objects: 100% (4/4), 447 bytes | 447.00 KiB/s, done.

Total 4 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/Suraj1808/Git_Exercise3.git
fe15051..aceac48 master -> master

Same error which came while me being a runner-up:

Suraj.Dalvi@PUN-DE-DL6VD02 MINGW64 ~/desktop/exercises (master)
\$ git push
To https://github.com/Suraj1808/Git_Exercise3.git
! [rejected] master -> master (non-fast-forward)
error: failed to push some refs to 'https://github.com/Suraj1808/Git_Exercise3.git'
hint: Updates were rejected because the tip of your current branch is behind
hint: its remote counterpart. Integrate the remote changes (e.g.
hint: 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.

Now changing the same files and then doing race:

\$ git status -s
M index.html

Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd
August/Suraj_Github_Repo/Git_Exercise3/Git_Exercise3/src (master)
\$ git add .

Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd
August/Suraj_Github_Repo/Git_Exercise3/Git_Exercise3/src (master)
\$ git commit -m 'Me is always a winner'
[master f973cd3] Me is always a winner
1 file changed, 1 insertion(+)

Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd
August/Suraj_Github_Repo/Git_Exercise3/Git_Exercise3/src (master)
\$ git push
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 8 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 454 bytes | 454.00 KiB/s, done.
Total 4 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/Suraj1808/Git_Exercise3.git
aceac48..f973cd3 master -> master

The screenshot shows the Git GUI interface. On the left, a commit history graph shows the progression from the first commit to the current master branch. The main panel displays a list of commits, with the most recent one highlighted. The bottom panel shows the details of the selected commit, including the author, committer, parent, child, branch, and follow-up information.

Commit history (from bottom to top):

- first commit
- second commit
- commit after 2 changes at staged and unstaged
- extra commit
- commit after remove cmd
- commit after change
- 1st commit in branching
- 2nd commit in branching
- 3rd commit in branching
- commit in master after merge
- commit in master
- b1 commit
- commit in b1
- commit done while being a runner-up
- this is commit for 7th task
- Merge branch 'master' of https://github.com/Suraj1808/Git_Exercise3
- Suraj's commit done
- Committing being a winner in the race to push first to the Github Repo
- suraj commit on same file
- remotes/origin/master
- master

Commit details (SHA1 ID: 754a188911c19079249eaeafb86f85824f16010a):

- Author: Suraj.Dalvi <Suraj.Dalvi@emtecinc.com> 2022-08-22 01:30:50
- Committer: Suraj.Dalvi <Suraj.Dalvi@emtecinc.com> 2022-08-22 01:30:50
- Parent: [aceac48161d1f096fd40c9cc892c44018ab8ea8d](#) (Committing being a winner i
- Child: [24d7c059b674afc57c5c17d3ccbe39519bae77f4](#) (Committed when we changed :
- Branch: [master](#)
- Follows:
- Precedes:

Exercise 4 – Main Task

1. Make a commit, and make a silly typo in the commit message.
2. Use the --amend flag to enable you to fix the commit message.
3. Look at the log and notice how the mistake is magically gone.

```
$ git push
```

To https://github.com/Suraj1808/Git_Exercise3.git

```
! [rejected] master -> master (non-fast-forward)
```

```
error: failed to push some refs to 'https://github.com/Suraj1808/Git_Exercise3.git'
```

hint: Updates were rejected because the tip of your current branch is behind

hint: its remote counterpart. Integrate the remote changes (e.g.

hint: 'git pull ...') before pushing again.

hint: See the 'Note about fast-forwards' in 'git push --help' for details.

Suraj.Dalvi@PUN-DE-DL6VD02 MINGW64 ~/desktop/exercises (master)

```
$ git pull
```

Auto-merging src/index.html

CONFLICT (content): Merge conflict in src/index.html

Automatic merge failed; fix conflicts and then commit the result.

Suraj.Dalvi@PUN-DE-DL6VD02 MINGW64 ~/desktop/exercises (master|MERGING)

\$ git status

On branch master

Your branch and 'origin/master' have diverged,
and have 1 and 1 different commits each, respectively.
(use "git pull" to merge the remote branch into yours)

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: src/index.html

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

Suraj.Dalvi@PUN-DE-DL6VD02 MINGW64 ~/desktop/exercises (master|MERGING)

\$ git status -s

UU src/index.html

Suraj.Dalvi@PUN-DE-DL6VD02 MINGW64 ~/desktop/exercises (master|MERGING)

\$ git add .

Suraj.Dalvi@PUN-DE-DL6VD02 MINGW64 ~/desktop/exercises (master|MERGING)

\$ git commit -m 'Committed when we changed same files after resolving the conflict'
[master 24d7c05] Committed when we changed same files after resolving the conflict

Suraj.Dalvi@PUN-DE-DL6VD02 MINGW64 ~/desktop/exercises (master)

\$ gitk

\$ git log

commit 0837f072a728bdace5fb71844b30a1cf92650530 (HEAD -> master)

Author: Suraj.Dalvi <Suraj.Dalvi@emtecinc.com>

Date: Mon Aug 22 01:41:57 2022 -0700

ABBBBBBBBBBBBBBB

\$ git commit --amend -m 'Fixing my typo commit message'

[master 6006fdc] Fixing my typo commit message

Date: Mon Aug 22 01:41:57 2022 -0700

1 file changed, 1 insertion(+)

\$ git log

commit 6006fdc35def207604439a8be74960c75c98bbc2 (HEAD -> master)

Author: Suraj.Dalvi <Suraj.Dalvi@emtecinc.com>

Date: Mon Aug 22 01:41:57 2022 -0700

Fixing my typo commit message

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

For using --amend with files we have to first put in the staging area first and then only we can do anything on them using --amend

5. Create a branch. Make a commit.

```
$ git checkout -b NewBranch
```

```
Switched to a new branch 'NewBranch'
```

```
Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd  
August/Suraj_Github_Repo/Git_Exercise3/Git_Exercise3 (NewBranch)
```

```
$ git branch
```

```
* NewBranch
```

```
master
```

```
Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd  
August/Suraj_Github_Repo/Git_Exercise3/Git_Exercise3 (NewBranch)
```

```
$ git status -s
```

```
M src/index.html
```

```
Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd  
August/Suraj_Github_Repo/Git_Exercise3/Git_Exercise3 (NewBranch)
```

```
$ git add .
```

```
Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd  
August/Suraj_Github_Repo/Git_Exercise3/Git_Exercise3 (NewBranch)
```

```
$ git commit -m 'First commit after creating the branch'
```

```
[NewBranch e990f44] First commit after creating the branch
```

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

```
Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd  
August/Suraj_Github_Repo/Git_Exercise3/Git_Exercise3 (NewBranch)
```

```
$ git status -s
```

```
Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd  
August/Suraj_Github_Repo/Git_Exercise3/Git_Exercise3 (NewBranch)
```

```
$ git log
```

```
commit e990f44b54b68ddb8b6a7194055b95be804ee2b (HEAD -> NewBranch)
```

```
Author: Atik Shaikh <atik.shaikh@emtecinc.com>
```

```
Date: Mon Aug 22 17:17:41 2022 +0530
```

First commit after creating the branch

commit f973cd36160215ade1e0d3d91cad384f3834ce59 (origin/master, origin/HEAD, master)

Author: Atik Shaikh <atik.shaikh@emtecinc.com>

Date: Mon Aug 22 14:00:20 2022 +0530

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

```
$ git checkout master
```

```
Switched to branch 'master'
```

```
Your branch is up to date with 'origin/master'.
```

```
$ git status -s
```

```
M src/file.html
```

```
Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd
```

```
August/Suraj_Github_Repo/Git_Exercise3/Git_Exercise3 (master)
```

```
$ git add .
```

```
Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd
```

```
August/Suraj_Github_Repo/Git_Exercise3/Git_Exercise3 (master)
```

```
$ git status -s
```

```
M src/file.html
```

```
Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd
```

```
August/Suraj_Github_Repo/Git_Exercise3/Git_Exercise3 (master)
```

```
$ git commit -m 'Committing while being on the master branch and it is non-conflicting commit'
[master 060e979] Committing while being on the master branch and it is non-conflicting commit
1 file changed, 1 insertion(+)
```

```
Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd
```

```
August/Suraj_Github_Repo/Git_Exercise3/Git_Exercise3 (master)
```

```
$ git status -s
```

```
Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd
```

```
August/Suraj_Github_Repo/Git_Exercise3/Git_Exercise3 (master)
```

```
$ git log
```

```
commit 060e979e1d1c3bb26c748d7e8cda7dcedae3b6ec (HEAD -> master)
```

```
Author: Atik Shaikh <atik.shaikh@emtecinc.com>
```

```
Date: Mon Aug 22 17:21:05 2022 +0530
```

Committing while being on the master branch and it is non-conflicting commit

commit f973cd36160215ade1e0d3d91cad384f3834ce59 (origin/master, origin/HEAD)

Author: Atik Shaikh <atik.shaikh@emtecinc.com>

Date: Mon Aug 22 14:00:20 2022 +0530

Me is always a winner

7. Now switch back to your branch.

```
$ git checkout NewBranch
```

```
Switched to branch 'NewBranch'
```

```
Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd  
August/Suraj_Github_Repo/Git_Exercise3/Git_Exercise3 (NewBranch)
```

```
$ git status -s
```

```
Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd  
August/Suraj_Github_Repo/Git_Exercise3/Git_Exercise3 (NewBranch)
```

```
$ git branch
```

```
* NewBranch
```

```
Master
```

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 branch
```

```
* NewBranch
```

```
master
```

```
Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd  
August/Suraj_Github_Repo/Git_Exercise3/Git_Exercise3 (NewBranch)
```

```
$ git rebase master
```

```
Successfully rebased and updated refs/heads/NewBranch.
```

```
Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd  
August/Suraj_Github_Repo/Git_Exercise3/Git_Exercise3 (NewBranch)
```

```
$ gitk
```



```
Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd
August/Suraj_Github_Repo/Git_Exercise3/Git_Exercise3 (NewBranch)
$ git log
commit d9b432e2a2dd7aa56507690e5719875fbfb75868 (HEAD -> NewBranch)
Author: Atik Shaikh <atik.shaikh@emtecinc.com>
Date: Mon Aug 22 17:31:13 2022 +0530
```

Committing after rebasing our NewBranch with master branch

```
commit 7863e0bc0e6abcef513d651b814dd53398adc9d1
Author: Atik Shaikh <atik.shaikh@emtecinc.com>
Date: Mon Aug 22 17:17:41 2022 +0530
```

First commit after creating the branch

```
commit 060e979e1d1c3bb26c748d7e8cda7dcedae3b6ec (master)
Author: Atik Shaikh <atik.shaikh@emtecinc.com>
Date: Mon Aug 22 17:21:05 2022 +0530
```

Committing while being on the master branch and it is non-conflicting commit

```
commit f973cd36160215ade1e0d3d91cad384f3834ce59 (origin/master, origin/HEAD)
Author: Atik Shaikh <atik.shaikh@emtecinc.com>
Date: Mon Aug 22 14:00:20 2022 +0530
```

Me is always a winner

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

You have two options for integrating your feature into the main branch: merging directly or rebasing and then merging.

```
$ git checkout master
Switched to branch 'master'
Your branch is ahead of 'origin/master' by 1 commit.
(use "git push" to publish your local commits)
```

```
Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd
August/Suraj_Github_Repo/Git_Exercise3/Git_Exercise3 (master)
$ git merge NewBranch
Updating 060e979..d9b432e
Fast-forward
 src/Atik.html | 2 +-
 src/file.html | 1 +
 src/index.html | 2 ++
 3 files changed, 4 insertions(+), 1 deletion(-)
```

Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd
 August/Suraj_Github_Repo/Git_Exercise3/Git_Exercise3 (master)
 \$ gitk

The screenshot shows the Git GUI (gitk) interface. The top menu bar includes File, Edit, View, and Help. The left pane displays a commit history graph with branches 'NewBranch' and 'master'. The right pane shows a table of commits with columns for author, commit message, and timestamp. The bottom pane provides a detailed view of the selected commit (SHA1 ID: d9b432e2a2dd7aa56507690e5719875fbfb75868), including the author (Atik Shaikh), committer (Atik Shaikh), parent (7863e0bc0e6abcef513d651b814dd53398adc9d1), and the commit message 'Committing after rebasing our NewBranch with master branch'. The commit message is followed by a diff showing changes to 'src/Atik.html'.

Author	Commit Message	Timestamp
Atik Shaikh <atik.shaikh@emtecinc.com>	Committing after rebasing our NewBranch with master branch	2022-08-22 17:31:13
Atik Shaikh <atik.shaikh@emtecinc.com>	Committing while being on the master branch and it is non-conflicting commit	2022-08-22 17:21:05
Atik Shaikh <atik.shaikh@emtecinc.com>	Committing being a winner in the race to push first to the Github Repo	2022-08-22 14:00:20
Atik Shaikh <atik.shaikh@emtecinc.com>	Suraj's commit done	2022-08-22 13:54:17
Suraj.Dalvi <Suraj.Dalvi@emtecinc.com>	Merge branch 'master' of https://github.com/Suraj1808/Git_Exercise3	2022-08-22 13:48:31
Atik Shaikh <atik.shaikh@emtecinc.com>	this is commit for 7th task	2022-08-22 13:35:40
Atik Shaikh <atik.shaikh@emtecinc.com>	Commit done while being a runner-up	2022-08-22 13:31:59
Atik Shaikh <atik.shaikh@emtecinc.com>	making changes in surajs github repo	2022-08-22 13:33:04
Atik Shaikh <atik.shaikh@emtecinc.com>	commit after conflict	2022-08-22 13:17:37
Suraj.Dalvi <Suraj.Dalvi@emtecinc.com>	commit in b1	2022-08-22 11:24:23
Suraj.Dalvi <Suraj.Dalvi@emtecinc.com>	b1 commit	2022-08-22 11:17:45
Suraj.Dalvi <Suraj.Dalvi@emtecinc.com>	commit in master	2022-08-22 11:16:12
Suraj.Dalvi <Suraj.Dalvi@emtecinc.com>	commit in master after merge	2022-08-22 11:20:18
Suraj.Dalvi <Suraj.Dalvi@emtecinc.com>	3rd commit in branching	2022-08-22 11:08:15
Suraj.Dalvi <Suraj.Dalvi@emtecinc.com>	2nd commit in branching	2022-08-22 10:56:20
Suraj.Dalvi <Suraj.Dalvi@emtecinc.com>	1st commit in branching	2022-08-22 10:55:29
Suraj.Dalvi <Suraj.Dalvi@emtecinc.com>	commit after change	2022-08-22 10:53:16
Suraj.Dalvi <Suraj.Dalvi@emtecinc.com>	commit after remove cmd	2022-08-22 10:18:58
Suraj.Dalvi <Suraj.Dalvi@emtecinc.com>	extra commit	2022-08-22 10:13:40
Suraj.Dalvi <Suraj.Dalvi@emtecinc.com>	commit after 2 changes at staged and unstaged	2022-08-22 10:07:32
Suraj.Dalvi <Suraj.Dalvi@emtecinc.com>	second commit	2022-08-22 09:58:20
Suraj.Dalvi <Suraj.Dalvi@emtecinc.com>	first commit	2022-08-22 09:48:04
Suraj.Dalvi <Suraj.Dalvi@emtecinc.com>		2022-08-22 09:44:52

SHA1 ID: d9b432e2a2dd7aa56507690e5719875fbfb75868

Find: commit containing:

Search

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

Author: Atik Shaikh <atik.shaikh@emtecinc.com> 2022-08-22 17:31:13
 Committer: Atik Shaikh <atik.shaikh@emtecinc.com> 2022-08-22 17:31:13
 Parent: 7863e0bc0e6abcef513d651b814dd53398adc9d1 (First commit after creati
 Branches: NewBranch, master
 Follows:
 Precedes:

Committing after rebasing our NewBranch with master branch

index 1b0780e..e4b6ac8 100644
 @@ -8,6 +8,6 @@

Stretch Task:

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

This is done at Suraj's Desk

2. Make a commit locally yourself also. Note that you should not have pulled their commit at this Point.
3. Try to push, and watch it fail

```
$ git status -s
M src/Atik.html
M src/file.html
```

Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd
 August/Suraj_Github_Repo/Git_Exercise3/Git_Exercise3 (master)


```
$ git add .
```

```
Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd
```

```
August/Suraj_Github_Repo/Git_Exercise3/Git_Exercise3 (master)
```

```
$ git commit -m 'Committing this one so as to demonstrate the use of --rebase flag in push'
```

```
[master 13bfc82] Committing this one so as to demonstrate the use of --rebase flag in push
```

```
2 files changed, 2 insertions(+)
```

```
Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd
```

```
August/Suraj_Github_Repo/Git_Exercise3/Git_Exercise3 (master)
```

```
$ git push
```

```
To https://github.com/Suraj1808/Git_Exercise3.git
```

```
! [rejected]      master -> master (fetch first)
```

```
error: failed to push some refs to 'https://github.com/Suraj1808/Git_Exercise3.git'
```

```
hint: Updates were rejected because the remote contains work that you do
```

```
hint: not have locally. This is usually caused by another repository pushing
```

```
hint: to the same ref. You may want to first integrate the remote changes
```

```
hint: (e.g., 'git pull ...') before pushing again.
```

```
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```

4. Now, pull but using the --rebase flag.

```
$ git pull --rebase
```

```
remote: Enumerating objects: 29, done.
```

```
remote: Counting objects: 100% (27/27), done.
```

```
remote: Compressing objects: 100% (8/8), done.
```

```
remote: Total 17 (delta 9), reused 17 (delta 9), pack-reused 0
```

```
Unpacking objects: 100% (17/17), 1.61 KiB | 2.00 KiB/s, done.
```

```
From https://github.com/Suraj1808/Git_Exercise3
```

```
  d9b432e..5eaff8e  master    -> origin/master
```

```
Successfully rebased and updated refs/heads/master.
```

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

```
$ git log
```

```
commit c3737b53c12179106b321684726d58820f81d33e (HEAD -> master, origin/master, origin/HEAD)
```

```
Author: Atik Shaikh <atik.shaikh@emtecinc.com>
```

```
Date: Mon Aug 22 17:46:59 2022 +0530
```

```
Committing this one so as to demonstrate the use of --rebase flag in push
```

```
commit 5eaff8e4f1b89acdc939a18f8af3d8debdeb8fd2
```

```
Merge: 6006fdc d9b432e
```

```
Author: Suraj.Dalvi <Suraj.Dalvi@emtecinc.com>
```

```
Date: Mon Aug 22 05:12:51 2022 -0700
```

```
Committing to resolve the conflict
```

```
commit d9b432e2a2dd7aa56507690e5719875fbfb75868 (NewBranch)
```

```
Author: Atik Shaikh <atik.shaikh@emtecinc.com>
```

```
Date: Mon Aug 22 17:31:13 2022 +0530
```

Committing after rebasing our NewBranch with master branch

Gitk

MINGW64/c:/Users/atik.shaikh/Desktop/Assignments/22nd August/Suraj_Github_Repo/Git_Exercise3/Git_Exercise3

Committing this one so as to demonstrate the use of --rebase flag in push

commit 5eaff8e4f1
Merge: 6006fdc d9
Author: Suraj.Dalvi
Date: Mon Aug 22 17:46:59 2022

Committing to
commit d9b432e2a2
Author: Atik Shaikh <atik.shaikh@emtecinc.com>
Date: Mon Aug 22 17:55:16 2022

Committing after
commit 7863e0bc0e
Author: Atik Shaikh <atik.shaikh@emtecinc.com>
Date: Mon Aug 22 17:46:59 2022

First commit
commit 060e979e1d
Author: Atik Shaikh <atik.shaikh@emtecinc.com>
Date: Mon Aug 22 17:46:59 2022

Fixing my typo
commit 6006fdc35d
Author: Suraj.Dalvi
Date: Mon Aug 22 17:46:59 2022

Atik.Shaikh@PUN-D
\$ gitk

Git_Exercise3: All files - gitk

File Edit View Help

SHA1 ID: c3737b53c12179106b321684726d58820f81d33e

Find: commit containing: Search

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

Author: Atik Shaikh <atik.shaikh@emtecinc.com> 2022-08-22 17:46:59
Committer: Atik Shaikh <atik.shaikh@emtecinc.com> 2022-08-22 17:55:16
Parent: 5eaff8e4f1b89acdc939a18f8af3d8debdeb8fd2 (Committing to resolve the conflict)
Branches: master, remotes/origin/master
Follows:
Precedes:

Committing this one so as to demonstrate the use of --rebase flag in push

Comments
src/Atik.html
src/file.html

Author	Commit	Date
Suraj.Dalvi <Suraj.Dalvi@emtecinc.com>	6006fdc35d	2022-08-22 17:46:59
Atik Shaikh <atik.shaikh@emtecinc.com>	5eaff8e4f1	2022-08-22 17:31:13
Atik Shaikh <atik.shaikh@emtecinc.com>	d9b432e2a2	2022-08-22 17:17:41
Atik Shaikh <atik.shaikh@emtecinc.com>	7863e0bc0e	2022-08-22 17:21:05
Suraj.Dalvi <Suraj.Dalvi@emtecinc.com>	d9b432e2a2	2022-08-22 14:11:57
Suraj.Dalvi <Suraj.Dalvi@emtecinc.com>	d9b432e2a2	2022-08-22 14:07:20
Atik Shaikh <atik.shaikh@emtecinc.com>	5eaff8e4f1	2022-08-22 14:00:20
Suraj.Dalvi <Suraj.Dalvi@emtecinc.com>	d9b432e2a2	2022-08-22 14:00:50
Atik Shaikh <atik.shaikh@emtecinc.com>	5eaff8e4f1	2022-08-22 13:54:17
Suraj.Dalvi <Suraj.Dalvi@emtecinc.com>	d9b432e2a2	2022-08-22 13:48:31
Atik Shaikh <atik.shaikh@emtecinc.com>	5eaff8e4f1	2022-08-22 13:35:40
Suraj.Dalvi <Suraj.Dalvi@emtecinc.com>	d9b432e2a2	2022-08-22 13:31:59
Atik Shaikh <atik.shaikh@emtecinc.com>	5eaff8e4f1	2022-08-22 13:33:04
Atik Shaikh <atik.shaikh@emtecinc.com>	5eaff8e4f1	2022-08-22 13:17:37
Suraj.Dalvi <Suraj.Dalvi@emtecinc.com>	d9b432e2a2	2022-08-22 11:24:23
Suraj.Dalvi <Suraj.Dalvi@emtecinc.com>	d9b432e2a2	2022-08-22 11:17:45
Suraj.Dalvi <Suraj.Dalvi@emtecinc.com>	d9b432e2a2	2022-08-22 11:16:12
Suraj.Dalvi <Suraj.Dalvi@emtecinc.com>	d9b432e2a2	2022-08-22 11:20:18
Suraj.Dalvi <Suraj.Dalvi@emtecinc.com>	d9b432e2a2	2022-08-22 11:08:15
Suraj.Dalvi <Suraj.Dalvi@emtecinc.com>	d9b432e2a2	2022-08-22 10:56:20
Suraj.Dalvi <Suraj.Dalvi@emtecinc.com>	d9b432e2a2	2022-08-22 10:55:29
Suraj.Dalvi <Suraj.Dalvi@emtecinc.com>	d9b432e2a2	2022-08-22 10:53:16
Suraj.Dalvi <Suraj.Dalvi@emtecinc.com>	d9b432e2a2	2022-08-22 10:18:58
Suraj.Dalvi <Suraj.Dalvi@emtecinc.com>	d9b432e2a2	2022-08-22 10:13:40

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

This is because of the fact that rebase flag always takes the latest commit one to the forward and merges it with previous commits.

SSH Key (works only for the SSH based remote URL):

So basically the use of SSH is that if I want to work on a PC without entering my username/email and password in it and then push/pull using that same pc then SSH key will act as an alternative for this same thing. There is nothing more to it than this.

So steps to gen a SSH key and then link it with the machine and the repository:

1. First go to <https://github.com/settings/keys>
2. Open Gitbash and after initializing an empty Git Repo enter this command (note it wont work before you init the repo)
`$ ssh-keygen -t ed25519 -C your_email@example.com`
3. After entering this command it would basically ask you to enter the file name stuff to store the pub/pvt SSH Keys.
4. Just keep pressing enter until the last prompt and you would get your private and public SSH keys generated on your system itself.
5. For this go to C:\Users\atik.shaikh\.ssh
You would see 2 Files (example.pub (Public SSH Key) and example(without any extension i.e it is private SSH)
Example.pub:
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIGpU4I7PCzwKlpZC+WO2vgeNEu7l8K3QaF01opcXA91N
atik.shaikh@emtecinc.com
6. Now you must go to your Github Repo and then link this private SSH key to basically generate a SSH Key
This SSH Key would basically act as an alternative to your traditional username/email and password which we would generally use while linking the local repo with the remote repo.
Eg:
Go to <https://github.com/settings/ssh/new>
Enter your example.pub(ssh-ed25519
AAAAC3NzaC1lZDI1NTE5AAAAIGpU4I7PCzwKlpZC+WO2vgeNEu7l8K3QaF01opcXA91N
atik.shaikh@emtecinc.com) key here and enter the title you want to give to it.
After doing this click on ADD SSH Key.
Once you do this you would get a SSH Key. (SHA256:PZPdmSi/dwt0fdLm+DH6SjJshe4HLUNPK5UXd1vXXc).
7. Create a private repository and there you would get a page which would give the command(`$git clone git@github.com:Emtecatikshaikh/Demo_SSH_Token.git`) to basically link your local repo with this Github repo but without username/email and password traditionally.
8. Once you get this command go to your Gitbash and execute this command there.
`$ git init`
Initialized empty Git repository in C:/Users/atik.shaikh/Desktop/Assignments/22nd August/SSH_Demo/.git/

Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd August/SSH_Demo (master)
`$ git clone git@github.com:Emtecatikshaikh/Demo_SSH_Token.git`
Cloning into 'Demo_SSH_Token'...
The authenticity of host 'github.com (20.207.73.82)' can't be established.
ED25519 key fingerprint is SHA256:+DiY3wvV6TuJJhbpZisF/zLDA0zPMSvHdkr4UvCOqU.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'github.com' (ED25519) to the list of known hosts.
warning: You appear to have cloned an empty repository.

The ssh key generated lets github verify you are the one pushing code and not someone else who might be malicious or otherwise. They are less cumbersome and more secure than manually entering a username/password as well. I'm not sure how you previously used github but SSH keys are best practice way to secure your repo so I would recommend you continue to use this key in the future.

Keys are basically decoupled passwords. Typically, there is a one-to-one mapping between passwords and accounts: each account has exactly one password, and each password is (or rather, should be) used with exactly one account.

However, any number of keys can be associated with a single account. This means you can have a different way of authenticating yourself from your personal computer, your work computer, your phone, your tablet, etc. If any one of those keys gets compromised, you can invalidate it without affecting the others.

This also allows for more secure account sharing: multiple maintainers can each have their own key for contributing to a shared repository

Personal Access Token:

1. Creating a personal access token

Go to <https://github.com/settings/apps>

2. Click on 'Generate new token'

3. Enter the Note (Title) and the expiration time, also select the scopes for this (Tick all the permissions)

4. Finally click on Generate Token.

5. PAT is generated and keep it copied somewhere cuz it is a secret password and shouldn't be shared with anyone else.

Use of the Personal Access Token (works only over HTTPS remote URL) thing:

Once you have a token, you can enter it instead of your password when performing Git operations over HTTPS.

Steps for it in the Gitbash:

1. \$ git clone <https://github.com/username/repo.git> (For Mac)

Username: your_username

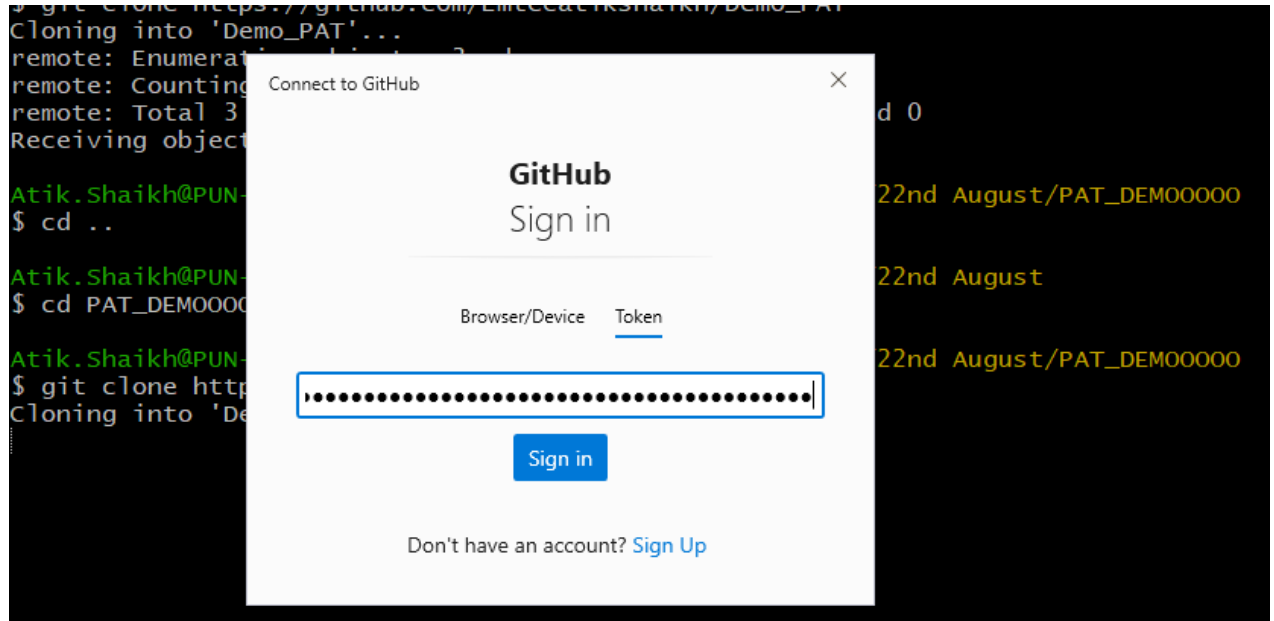
Password: your_token

2. After entering the username and the **Personal Access Token (Instead of Password)** , the username and token are successfully linked to the remote repo i.e the private Git Repo is linked to the local repo successfully without even using the username and password which you would use typically for linking the local repo with the Git Repo.

Suppose I have already have one locally cached username/token for 1 project and I want to switch from this one to another one how would I do that ?

- a. Install Github CLI
- b. gh auth login
 - When prompted for your preferred protocol for Git operations, select HTTPS.
 - When asked if you would like to authenticate to Git with your GitHub credentials, enter Y.

For Windows it basically gives a GUI where you can enter the PAT thing and then do the things from there on:



SSH v/s Personal Access Token (Both are valid only for Private Repositories only):

1. Personal access tokens can only be used for HTTPS Git operations. If your repository uses an SSH remote URL, you will need to switch the remote from SSH to HTTPS.
2. SSH is mostly for the servers
HTTPS (PAT thing) will work for mostly all the sites that starts with HTTPs (Internet basically)

Extra Github Commands:

Git Cherry Pick:

You can apply an existing commit from another branch to the current branch within the repository by using the git cherry-pick command. Cherry-picking allows you to:

- Move a commit that was committed to the wrong branch to the right branch.
- Add a commit to the current branch based on an existing commit from another branch.

Suppose we have to do a commit in master which was done in other branch. For this first I have to visit the master branch first and then do the cherrypick cmd.

```
$ git log
commit 3a1ec5889a49f01309c71b881dbc24575e98d462 (HEAD -> master)
Author: Atik Shaikh <atik.shaikh@emtecinc.com>
Date: Tue Aug 23 11:47:15 2022 +0530
```

This commit is done while being on the master branch

```
Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd August/Git_CherryPick
(master)
```

```
$ git cherry-pick 2130f2c
[master e990a75] This is the wrong commit for which I would use cherrypick command
Date: Tue Aug 23 11:48:58 2022 +0530
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 Second.html
```

```
Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd August/Git_CherryPick
(master)
```

```
$ git log
commit e990a757c90bb7d5a36a348b8e8065f9e8094cdb (HEAD -> master)
Author: Atik Shaikh <atik.shaikh@emtecinc.com>
Date: Tue Aug 23 11:48:58 2022 +0530
```

This is the wrong commit for which I would use cherrypick command

```
commit 3a1ec5889a49f01309c71b881dbc24575e98d462
Author: Atik Shaikh <atik.shaikh@emtecinc.com>
Date: Tue Aug 23 11:47:15 2022 +0530
```

This commit is done while being on the master branch

```
Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd August/Git_CherryPick
(master)
```

```
$ git reflog
e990a75 (HEAD -> master) HEAD@{0}: cherry-pick: This is the wrong commit for which I would use
cherrypick command
3a1ec58 HEAD@{1}: checkout: moving from Second_Branch to master
2130f2c (Second_Branch) HEAD@{2}: commit: This is the wrong commit for which I would use
cherrypick command
```

48d79d1 (New_Branch) HEAD@{3}: checkout: moving from New_Branch to Second_Branch
48d79d1 (New_Branch) HEAD@{4}: commit: This commit is done while being on the New_Branch
3a1ec58 HEAD@{5}: checkout: moving from master to New_Branch
3a1ec58 HEAD@{6}: commit (initial): This commit is done while being on the master branch

Git merge squash:

commit 73f89994bdb91ee0a38f4231e00fc8d247d7039c (HEAD -> master)
Author: Atik Shaikh <atik.shaikh@emtecinc.com>
Date: Tue Aug 23 12:10:27 2022 +0530

Squashed commit of the following:

commit 701b56545c569bbe7b139db3322bc464ca497713
Author: Atik Shaikh <atik.shaikh@emtecinc.com>
Date: Tue Aug 23 12:08:49 2022 +0530

XD_Merge-Squash

commit 3c0a839dfba9895b2a21486905dd14a94c6bb435
Author: Atik Shaikh <atik.shaikh@emtecinc.com>
Date: Tue Aug 23 12:00:19 2022 +0530

Trying something new

commit 3a1ec5889a49f01309c71b881dbc24575e98d462
Author: Atik Shaikh <atik.shaikh@emtecinc.com>
Date: Tue Aug 23 11:47:15 2022 +0530

This commit is done while being on the master branch

\$git reflog – To see the changes done overall from the start of initializing the repo

Best Practise to remove specific commits:

git rebase -i HEAD~x

(Note: x is the number of commits)

upon executing notepad file will open. Enter drop besides your commit.

If you don't know Vim, just click on each word pick that you want to edit and then hit the "I" key (for insert mode). Once you're done typing hit the "esc" key to exit insert mode.

--no-verify

So, this command is basically used to skip the git commit hooks. Alternative for the --no-verify is -n which is the short form for no verify.

Git Hooks:

Git hooks are scripts that automatically run every time a particular event occurs in a Git repository. They let you customize Git's internal behavior and trigger customizable actions at key points in the development life cycle.

Example:

git commit -m "commit message" --no-verify. When the --no-verify option is used, the pre-commit and commit-msg hooks are bypassed.

Patches:

A patch is a small file that indicates the changes made in a repository. It's generally used when someone from outside your team has read-only access but had a good code change available. He then creates a patch and sends it to you. You apply it and push it to the git repository. Everyone then benefits from the updated version, and the author of the patch didn't need read/write access.

```
$ touch one.html
```

```
Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd August/Git_CherryPick
(master)
```

```
$ touch two.html
```

```
Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd August/Git_CherryPick
(master)
```

```
$ git status -s
```

```
?? one.html
```

```
?? two.html
```

```
Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd August/Git_CherryPick
(master)
```

```
$ git add .
```

```
Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd August/Git_CherryPick
(master)
```

```
$ git diff --cached > mypatch.patch
```

```
Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd August/Git_CherryPick
(master)
```

```
$ cat mypatch.patch
```

```
diff --git a/one.html b/one.html
```

```
new file mode 100644
```

```
index 0000000..43bdb68
```

```
--- /dev/null
```



```

+++ b/one.html
@@ -0,0 +1,14 @@
+<!DOCTYPE html>
+<html lang="en">
+<head>
+  <meta charset="UTF-8">
+  <meta name="viewport" content="width=device-width, initial-scale=1.0">
+  <meta http-equiv="X-UA-Compatible" content="ie=edge">
+  <title>Document</title>
+</head>
+<body>
+  <h1>This line is same in both the files</h1>
+  <h2>This line isnt there in two.html</h2>
+
+</body>
+</html>

```

```

\ No newline at end of file
diff --git a/two.html b/two.html

```

```

new file mode 100644
index 0000000..6fe7d6f
--- /dev/null

```

```

+++ b/two.html
@@ -0,0 +1,12 @@
+<!DOCTYPE html>
+<html lang="en">
+<head>
+  <meta charset="UTF-8">
+  <meta name="viewport" content="width=device-width, initial-scale=1.0">
+  <meta http-equiv="X-UA-Compatible" content="ie=edge">
+  <title>Document</title>
+</head>
+<body>
+  <h1>This line is same in both the files</h1>
+</body>
+</html>

```

```

\ No newline at end of file

```

```

$ git status -s
A one.html
A two.html
?? mypatch.patch

```

```

$ git mv one.html 1.html

```

```

Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd August/Git_CherryPick
(NewBranch)

```

```

$ git mv two.html 2.html

```

```
Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd August/Git_CherryPick  
(NewBranch)
```

```
$ git add .
```

```
Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd August/Git_CherryPick  
(NewBranch)
```

```
$ git status -s
```

```
A 1.html
```

```
A 2.html
```

```
A mypatch.patch
```

```
A third.html
```

```
Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd August/Git_CherryPick  
(NewBranch)
```

```
$ git apply mypatch.patch
```

```
mypatch.patch:18: trailing whitespace.
```

```
warning: 1 line adds whitespace errors.
```

```
Atik.Shaikh@PUN-DE-LEFT88T MINGW64 ~/Desktop/Assignments/22nd August/Git_CherryPick  
(NewBranch)
```

```
$ git status -s
```

```
A 1.html
```

```
A 2.html
```

```
A mypatch.patch
```

```
A third.html
```

```
?? one.html
```

```
?? two.html
```

Git Fetch

git fetch is a primary command used to download contents from a remote repository. git fetch is used in conjunction with git remote, git branch, git checkout, and git reset to update a local repository to the state of a remote. The git fetch command is a critical piece of collaborative git work flows. git fetch has similar behavior to git pull, however, git fetch can be considered a safer, nondestructive version.