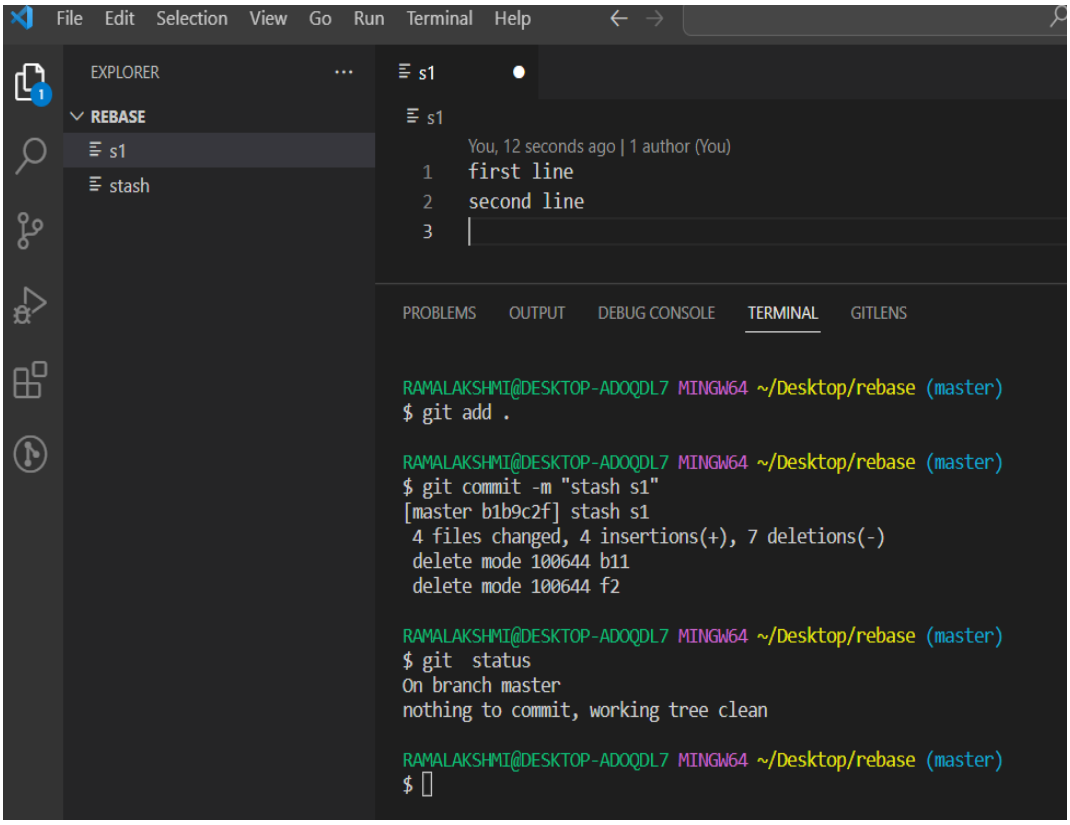
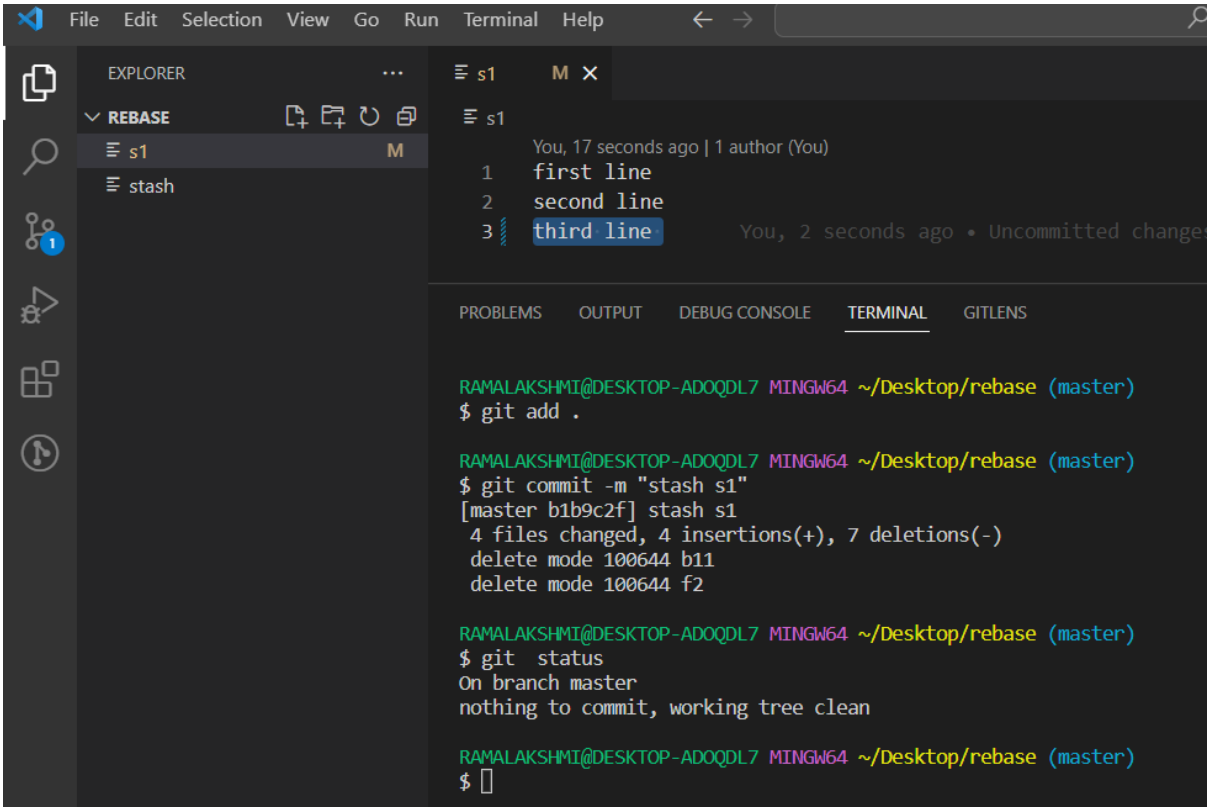
ASSIGNMENT:

1.GIT STASH:

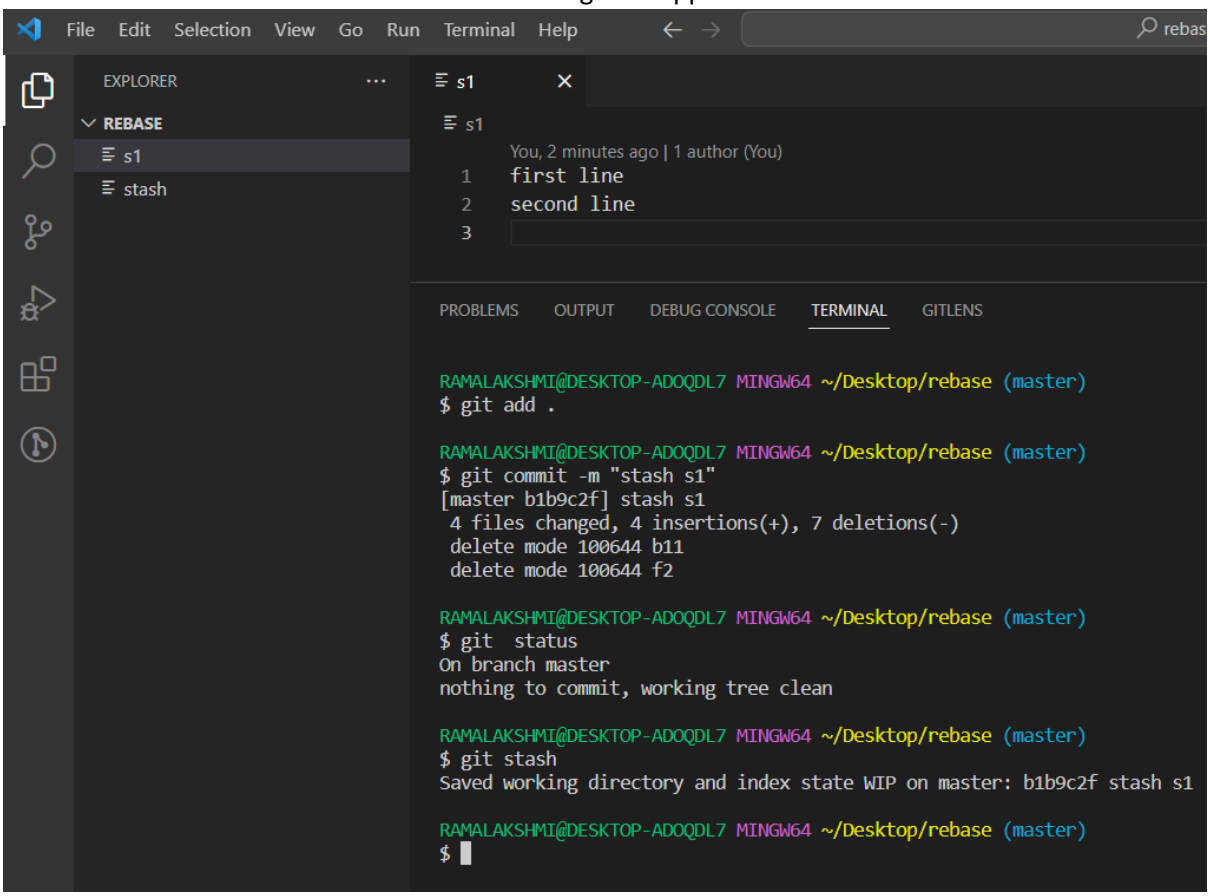
- git stash temporarily shelves (or stashes) changes you've made to your working copy so you can work on something else, and then come back and re-apply them later on.
- I have created a file namely **s1** and then added two lines.
- After that I have added the file and commited that as shown in below screenshot.



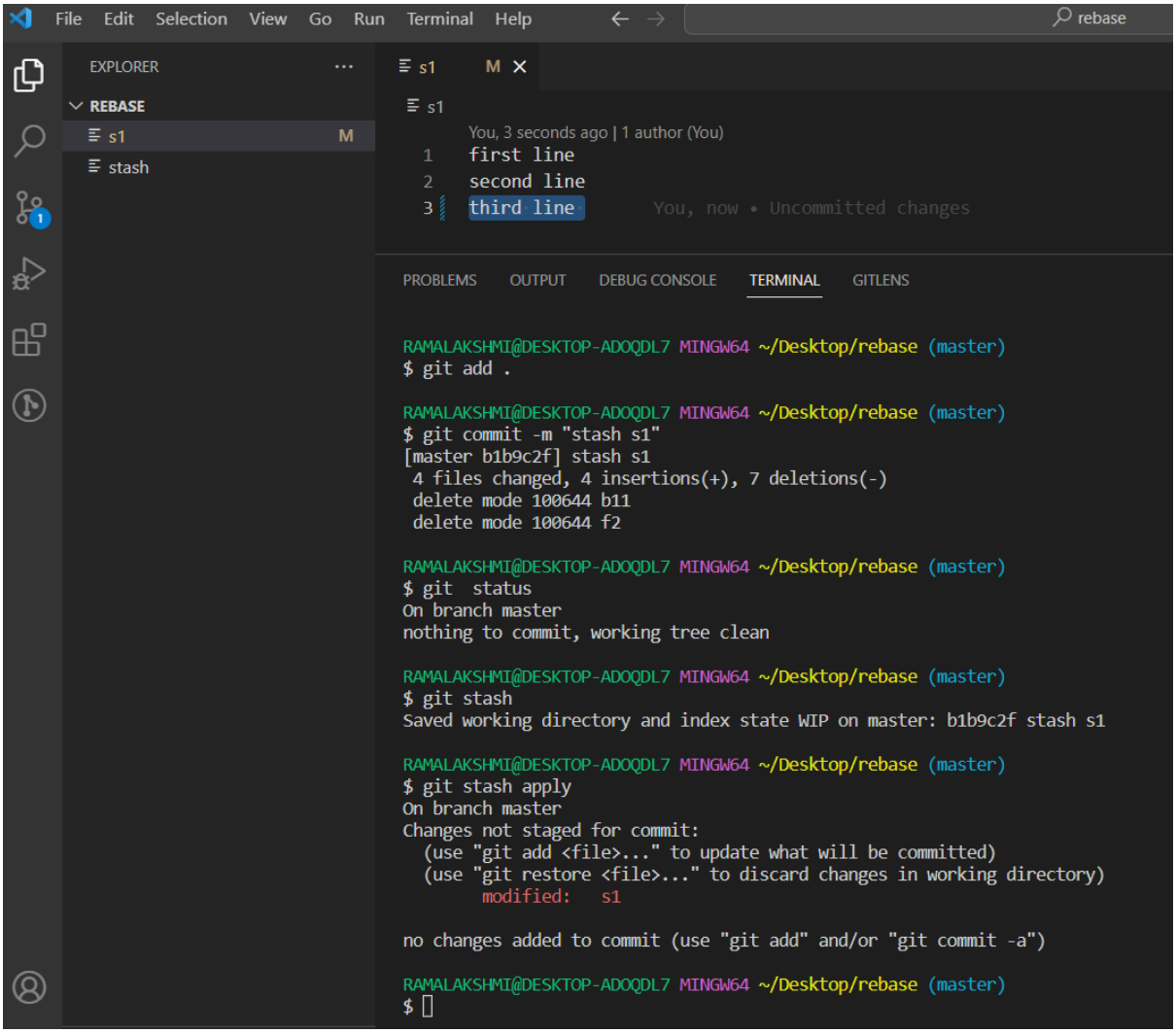
- Now I have added a third line and just saved it ,but did not add or commit it.



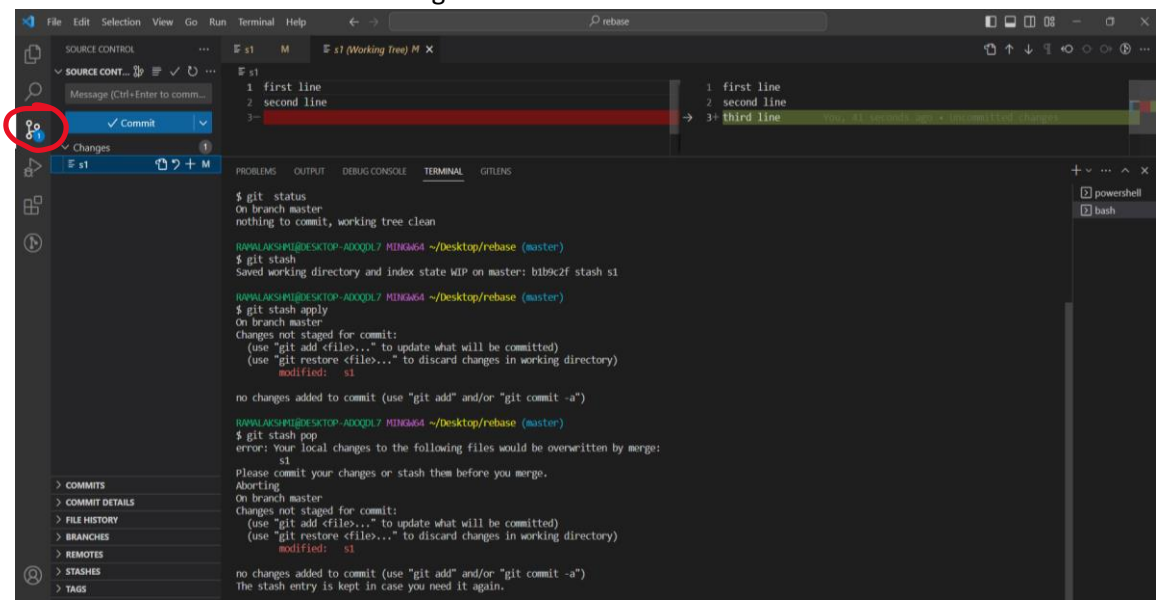
- So here after I have stashed file so the third line got disapperared since we didn't commit it.



➢ The stashed content can be obtained by using the **git stash apply** .



➢ Shows us the stashed as well as original data as shown below.



- **Git stash list** gives us the list of stashed commits.

```
RAMALAKSHMI@DESKTOP-ADOQDL7 MINGW64 ~/Desktop/rebase (master)
$ git stash list
stash@{0}: WIP on master: b1b9c2f stash s1
stash@{1}: WIP on master: b7ec72d s22
stash@{2}: WIP on master: bff69d4 added
stash@{3}: WIP on master: 41873ec second stash
stash@{4}: WIP on master: 1fdb251 stash commit
```

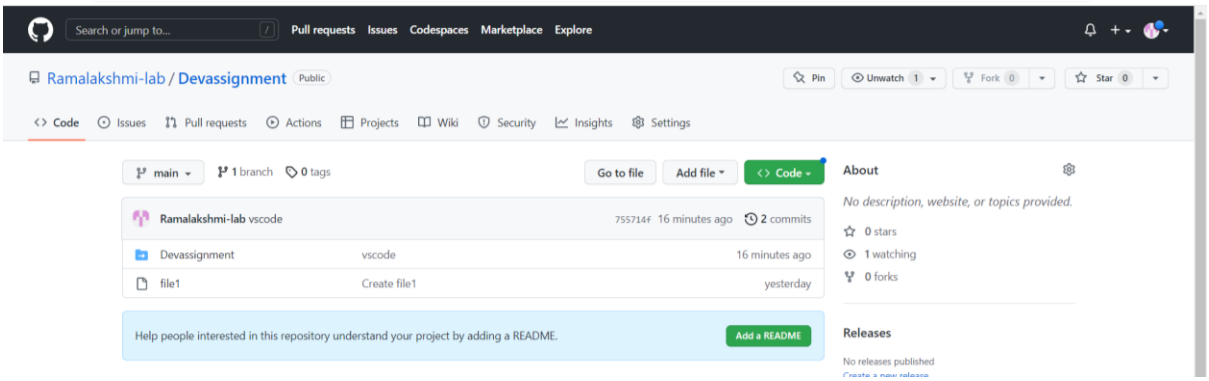
2.GIT FETCH and Merge:

Use of Git Fetch and Git Merge Command:

Git Fetch is used to bring the Remote changes to the Local Repository

For example, we have created a file in the Repository locally and push that file into the Repository. We made some changes in that file Remotely and committed there. So to get that changes Locally we Fetch those Changes by using the Fetch Command. To observe the changes we will merge the branches.

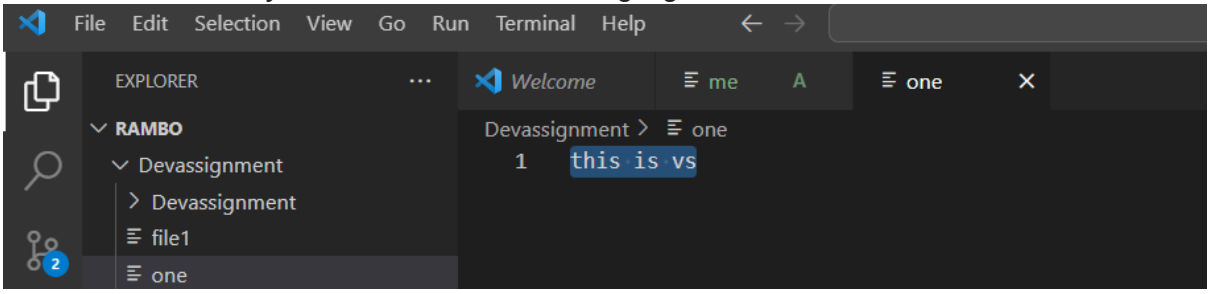
- I am going to clone a repository named **Devassignment.**
- U can see the repository before further filles addition.



- I am changing to that repository and then create a file

```
RAMALAKSHMI@DESKTOP-ADOQDL7 MINGW64 ~/Desktop/rambo (master)
$ git clone "https://github.com/Ramalakshmi-lab/Devassignment"
Cloning into 'Devassignment'...
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 5 (delta 0), reused 2 (delta 0), pack-reused 0
Receiving objects: 100% (5/5), done.

RAMALAKSHMI@DESKTOP-ADOQDL7 MINGW64 ~/Desktop/rambo (master)
$ cd Devassignment
```

- Created a file namely one and inserted a line highlighted below.



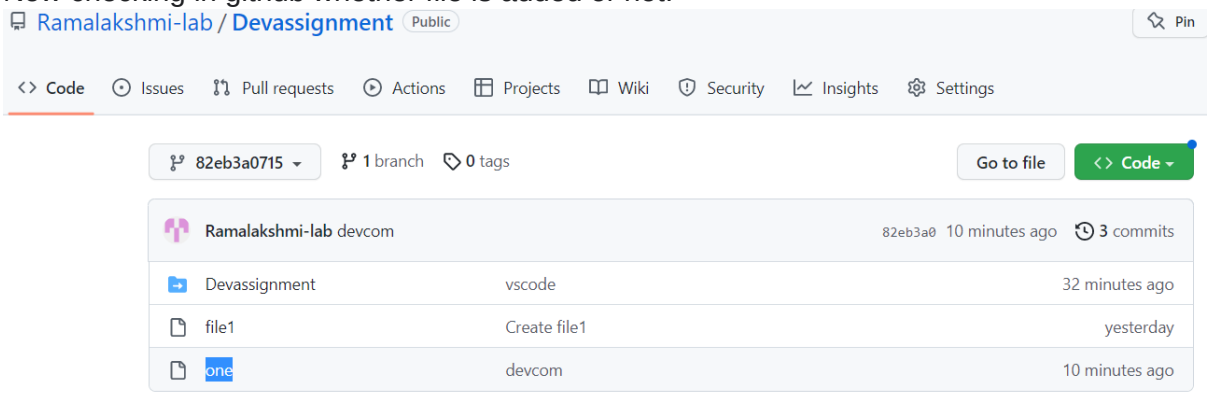
- I am going to add and commit file.

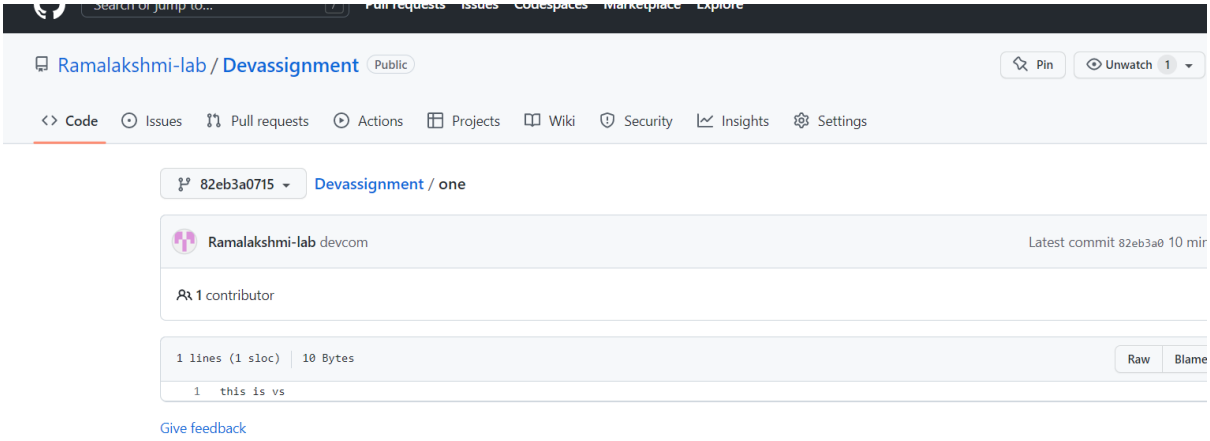
```
RAMALAKSHMI@DESKTOP-ADOQDL7 MINGW64 ~/Desktop/rambo/Devassignment (main)
$ git add .

RAMALAKSHMI@DESKTOP-ADOQDL7 MINGW64 ~/Desktop/rambo/Devassignment (main)
$ git commit -m "devcom"
[main 82eb3a0] devcom
 1 file changed, 1 insertion(+)
 create mode 100644 one
```

- Now I am going to push file.

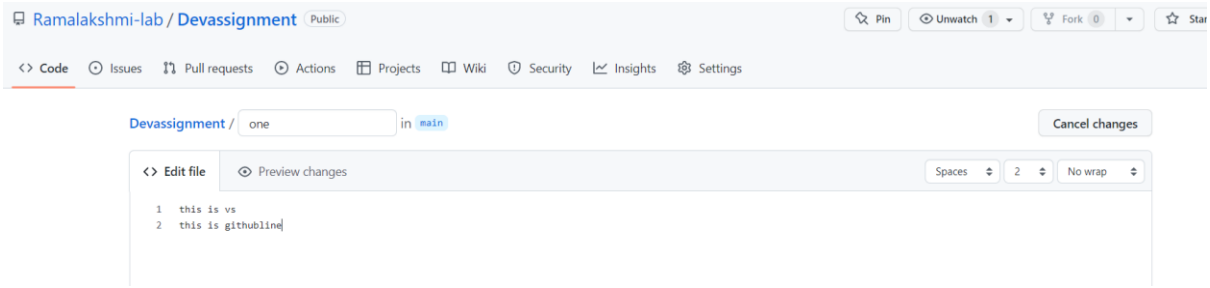
```
RAMALAKSHMI@DESKTOP-ADOQDL7 MINGW64 ~/Desktop/rambo/Devassignment (main)
$ git push
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 304 bytes | 152.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/Ramalakshmi-lab/Devassignment
   755714f..82eb3a0  main -> main
```

- Now checking in github whether file is added or not.





After editing remotely



- Now fetch it.

```
RAMALAKSHMI@DESKTOP-ADOQDL7 MINGW64 ~/Desktop/rambo/Devassignment (main)
$ git fetch
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 653 bytes | 29.00 KiB/s, done.
From https://github.com/Ramalakshmi-lab/Devassignment
   82eb3a0..41cb638  main       -> origin/main
```

- Now merge it.

```
RAMALAKSHMI@DESKTOP-ADOQDL7 MINGW64 ~/Desktop/rambo/Devassignment (main)
$ git merge origin/main
Updating 82eb3a0..41cb638
Fast-forward
 one | 3 ++-
 1 file changed, 2 insertions(+), 1 deletion(-)
```

- Observe the changes.

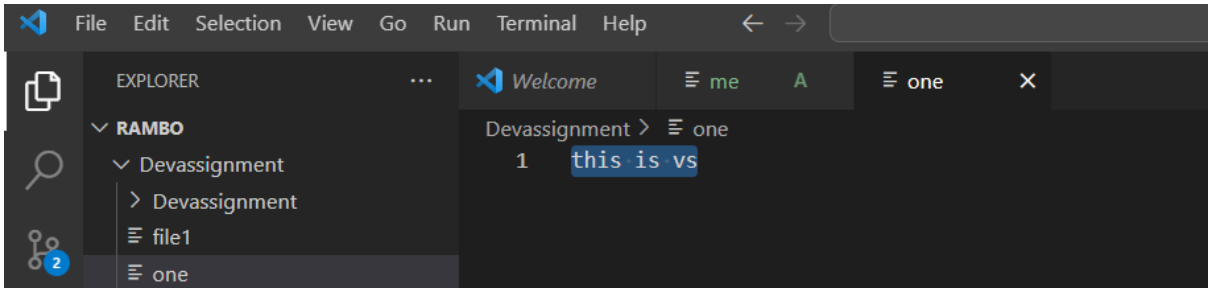
```
RAMALAKSHMI@DESKTOP-ADOQDL7 MINGW64 ~/Desktop/rambo/Devassignment (main)
$ git log
commit 41cb63815927a368f6e4f33732d7d0098f38d8f6 (HEAD -> main, origin/main, origin/HEAD)
Author: 20A91A0579 <84506201+Ramalakshmi-lab@users.noreply.github.com>
Date:   Fri Feb 17 22:11:26 2023 +0530

    Update one
```

- Since we have fetched file and merged the remoted changes are also reflected here Let us just use cat command to verify whether changes are reflected or not.

```
RAMALAKSHMI@DESKTOP-ADOQDL7 MINGW64 ~/Desktop/rambo/Devassignment (main)
$ cat one
this is vs
this is githubline
```

Finally

Before :



After:

```
RAMALAKSHMI@DESKTOP-ADOQDL7 MINGW64 ~/Desktop/rambo/Devassignment (main)
$ cat one
this is vs
this is githubline
```

3. Difference between Git Fetch and Git Pull:

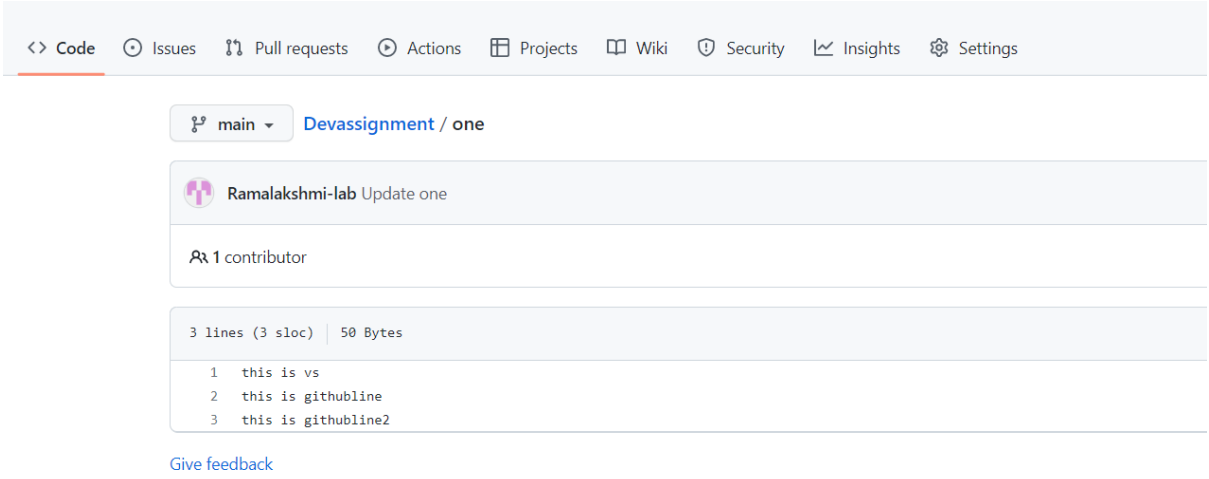
Git Fetch is the command that tells the local repository that there are changes available in the remote repository without bringing the changes into the local repository.

Git Pull on the other hand brings the copy of the remote directory changes into the local repository.

As we have seen the Git Fetch above, now let's see what is the difference between Git Fetch and Git Pull

In Git Fetch Command, first we have to fetch the changes and then merge them. But Git pull is used to Fetch and merge the changes at the same time. Instead of Giving Fetch and Merge Commands separately, we can use the pull Command

- I have again added another line remotedly to the one file in Devassignment Repository as shown below.



- Using a pull request.

```
RAMALAKSHMI@DESKTOP-ADOQDL7 MINGW64 ~/Desktop/rambo/Devassignment (main)
$ git pull origin main
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 658 bytes | 27.00 KiB/s, done.
From https://github.com/Ramalakshmi-lab/Devassignment
 * branch            main       -> FETCH_HEAD
   41cb638..71679be  main       -> origin/main
Updating 41cb638..71679be
Fast-forward
 one | 1 +
 1 file changed, 1 insertion(+)
```

- Checking whether changes are updated.

```
RAMALAKSHMI@DESKTOP-ADOQDL7 MINGW64 ~/Desktop/rambo/Devassignment (main)
$ git log
commit 71679bec5f56066b7477385647cdef60066175b0 (HEAD -> main, origin/main, origin/HEAD)
Author: 20A91A0579 <84506201+Ramalakshmi-lab@users.noreply.github.com>
Date:   Fri Feb 17 22:24:26 2023 +0530

    Update one

commit 41cb63815927a368f6e4f33732d7d0098f38d8f6
Author: 20A91A0579 <84506201+Ramalakshmi-lab@users.noreply.github.com>
Date:   Fri Feb 17 22:11:26 2023 +0530
```

- Using cat command to check whether the pull was successful or not.

```
RAMALAKSHMI@DESKTOP-ADOQDL7 MINGW64 ~/Desktop/rambo/Devassignment (main)
$ cat one
this is vs
this is githubline
this is githubline2
```

Finally:
Before

```
RAMALAKSHMI@DESKTOP-ADOQDL7 MINGW64 ~/Desktop/rambo/Devassignment (main)
$ cat one
this is vs
this is githubline
```

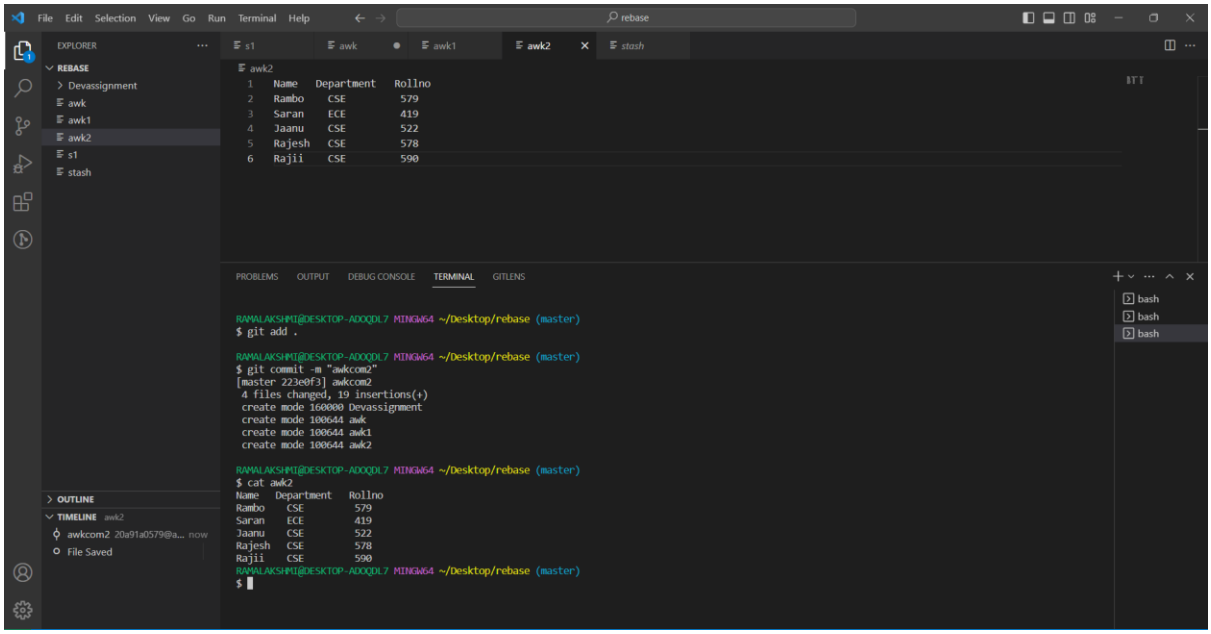
After:

```
RAMALAKSHMI@DESKTOP-ADOQDL7 MINGW64 ~/Desktop/rambo/Devassignment (main)
$ cat one
this is vs
this is githubline
this is githubline2
```

4.AWK COMMAND:

AWK is used for Pattern Scanning and Processing. It is used for Reading the Files. We can specify the patterns and fetch the data from the file. We can also count the number of input records and fields in the File.

- I have created file namely awk2 ,added and commited it and displayed it using cat command.



- We can use awk command to display file contents just like cat does.

```
RAMALAKSHMI@DESKTOP-ADOQDL7 MINGW64 ~/Desktop/rebase (master)
$ awk '{print}' awk2
Name   Department   Rollno
Rambo    CSE          579
Saran    ECE          419
Jaanu    CSE          522
Rajesh   CSE          578
Rajii    CSE          590
```

- Below command gives us the entire row which contains something specified is present in that row (ex:CSE)

```
RAMALAKSHMI@DESKTOP-ADOQDL7 MINGW64 ~/Desktop/rebase (master)
$ awk '/CSE/ {print}' awk2
Rambo    CSE          579
Jaanu    CSE          522
Rajesh   CSE          578
Rajii    CSE          590
```

- We can display some columns by specifying them using $ followed by the column no just like command line arguments.

```
RAMALAKSHMI@DESKTOP-ADOQDL7 MINGW64 ~/Desktop/rebase (master)
$ awk '{print $1,$3}' awk2
Name Rollno
Rambo 579
Saran 419
Jaanu 522
Rajesh 578
Rajii 590
```

- Numbers can be given to rows using awk command just like below using **NR**.

```
RAMALAKSHMI@DESKTOP-ADOQDL7 MINGW64 ~/Desktop/rebase (master)
$ awk '{print NR,$0}' awk2
1 Name   Department   Rollno
2 Rambo    CSE         579
3 Saran    ECE         419
4 Jaanu    CSE         522
5 Rajesh   CSE         578
6 Rajii    CSE         590
```

- **NF** command counts the number of fields in an input Record, Here NF is considered as the last field

```
RAMALAKSHMI@DESKTOP-ADOQDL7 MINGW64 ~/Desktop/rebase (master)
$ awk '{print $1,$NF}' awk2
Name Rollno
Rambo 579
Saran 419
Jaanu 522
Rajesh 578
Rajii 590
```

- Prime number in between 1 .. 20 in bash script.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  GITLENS

for num in {1..20}; do
  prime=true
  for (( i=2; i<$num; i++ )); do
    if (( $num % $i == 0 )); then
      prime=false
      break
    fi
  done
  if [ $prime == true ]; then
    echo $num
  fi
done
~
```

```
RAMALAKSHMI@DESKTOP-ADOQDL7 MINGW64 ~/Desktop/rebase (master)
$ vi primee.sh

RAMALAKSHMI@DESKTOP-ADOQDL7 MINGW64 ~/Desktop/rebase (master)
$ bash primee.sh
1
2
3
5
7
11
13
17
19
```

Q5. Set up a container and run a Ubuntu operating system. For this purpose, you can make use of the docker hub and run the container in interactive mode. All the processes pertaining to this should be provided in a screenshot for grading.

- Keep Your docker connected to hub and perform following commands to setup your container and run a Ubuntu operating system