

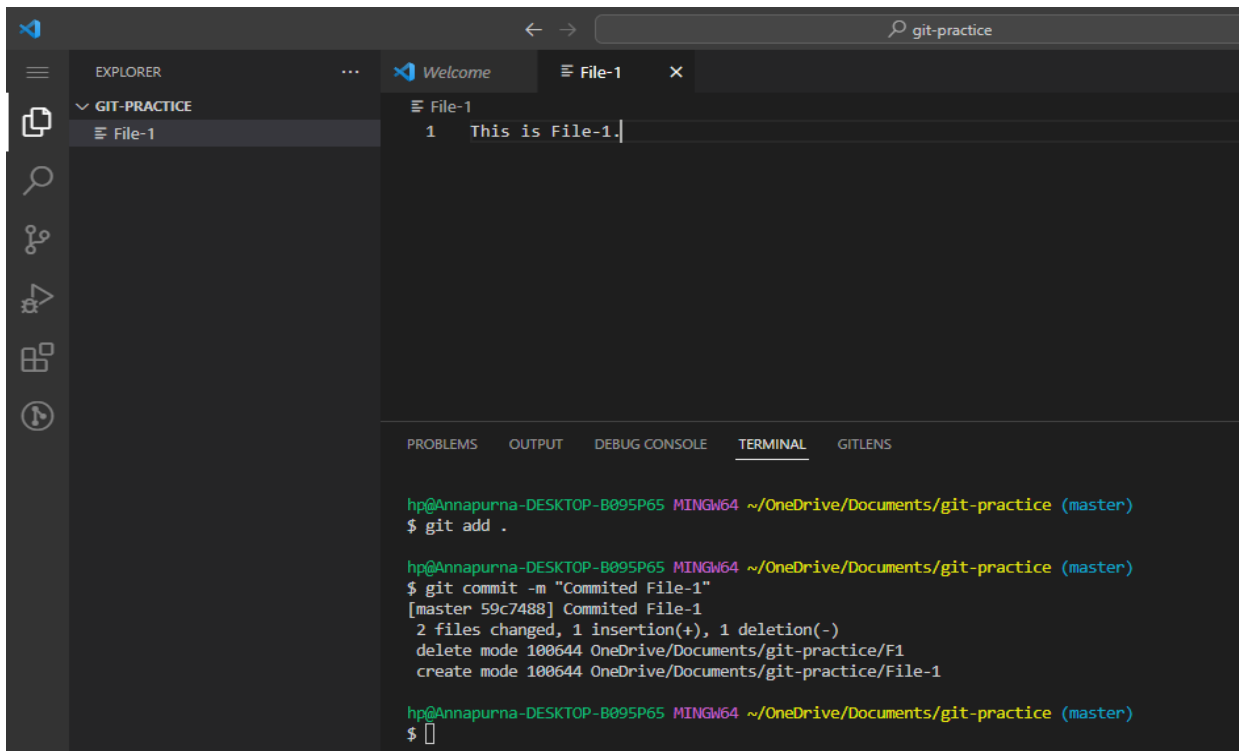
**Q1. Describe the usage of the git stash command by using an example and also state the process by giving the screenshot of all the commands written in git bash.**

## git stash

Git stash saves the uncommitted changes locally, allowing us to make changes, switch branches and can do all other git operations. We can reapply the stashed changes when we need them.

For example, if me and my friend are working with two branches A and B which are merged together. If suppose currently I am working in branch A with some file and I haven't committed the changes yet and my friend got a bug in the branch B and ask me to fix it so, when I switch to branch B then it will show me an error that "Your local changes to the following files would be overwritten by checkout ... Please commit your changes or stash them before you switch branches.". So, to overcome from this problem we use stash it stores the changes and we can retrieve them whenever we want.

- First I have created a file named File-1 and added some content in it. I have added it and committed it.

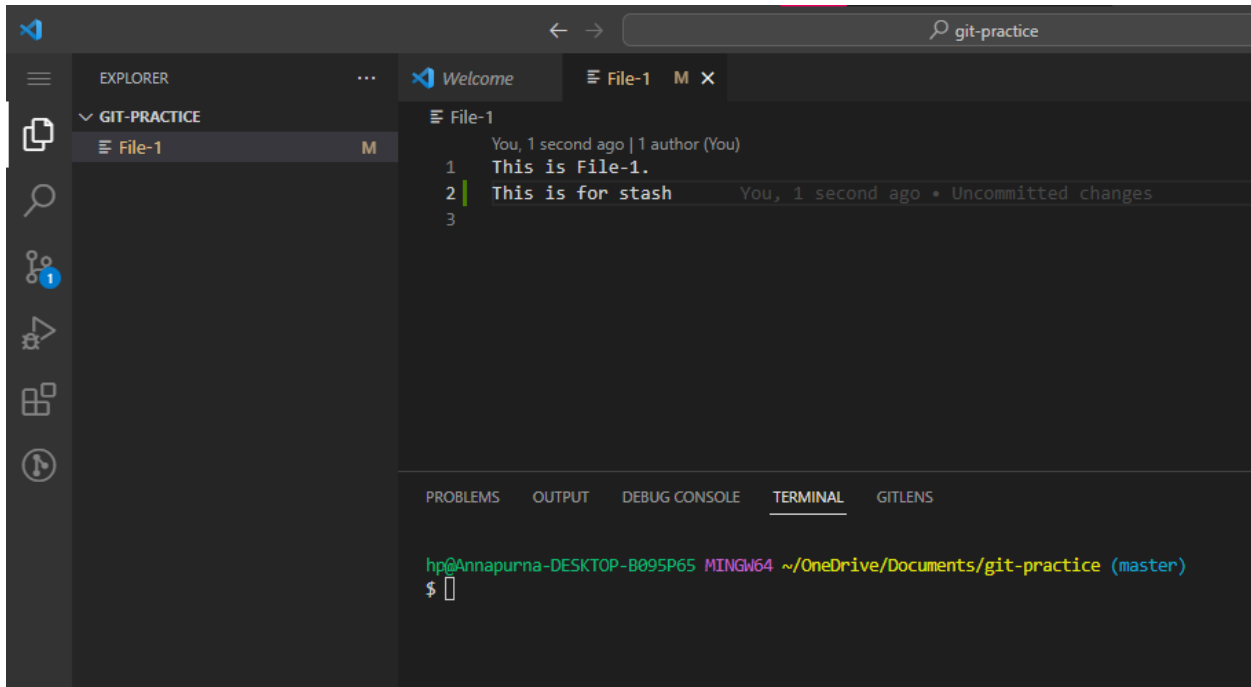


```
hp@Annapurna-DESKTOP-B095P65 MINGW64 ~/OneDrive/Documents/git-practice (master)
$ git add .

hp@Annapurna-DESKTOP-B095P65 MINGW64 ~/OneDrive/Documents/git-practice (master)
$ git commit -m "Committed File-1"
[master 59c7488] Committed File-1
2 files changed, 1 insertion(+), 1 deletion(-)
delete mode 100644 OneDrive/Documents/git-practice/F1
create mode 100644 OneDrive/Documents/git-practice/File-1

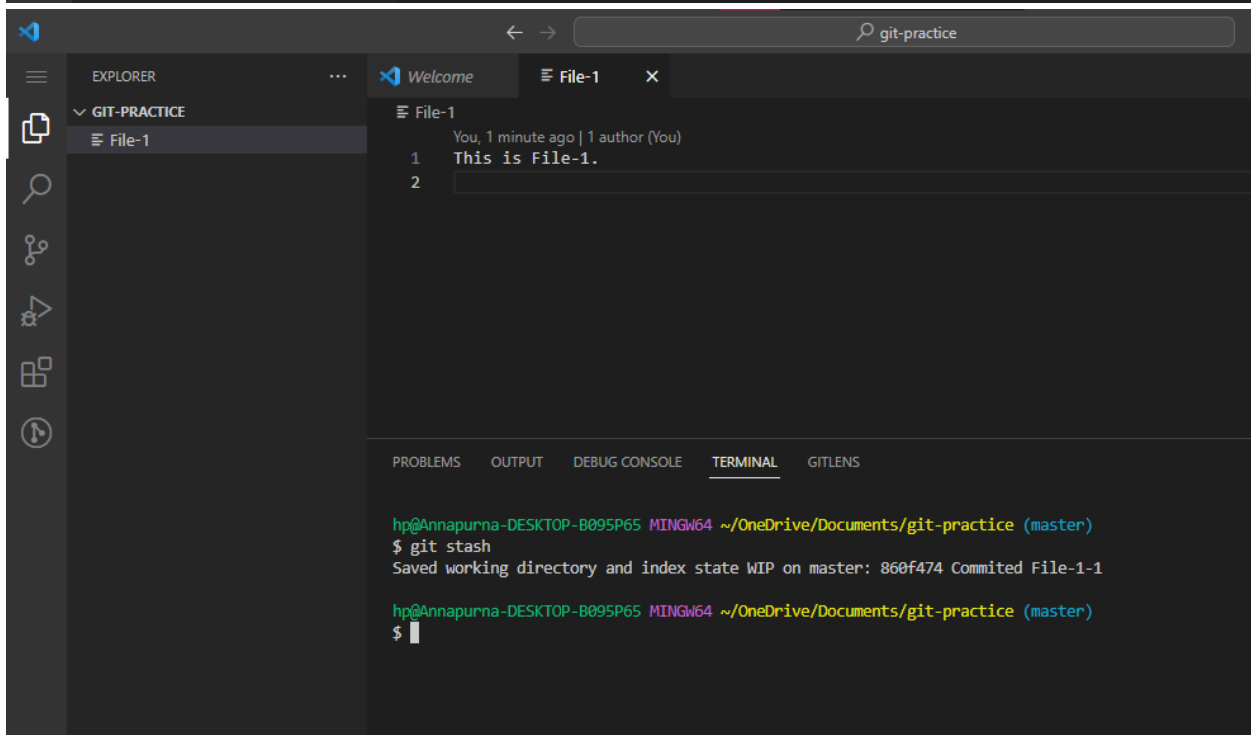
hp@Annapurna-DESKTOP-B095P65 MINGW64 ~/OneDrive/Documents/git-practice (master)
$
```

Now I have added another line in the File-1 and I have saved it.



The screenshot shows the Visual Studio Code interface with a file named 'File-1' open. The file contains two lines of code: 'This is File-1.' and 'This is for stash'. The second line is highlighted in green, and a status bar at the bottom right indicates 'You, 1 second ago • Uncommitted changes'. The terminal at the bottom shows the user's prompt and a cursor.

```
hp@Annapurna-DESKTOP-B095P65 MINGW64 ~/OneDrive/Documents/git-practice (master)
$
```



The screenshot shows the Visual Studio Code interface with 'File-1' open. Only the first line of code, 'This is File-1.', is visible. The terminal at the bottom shows the execution of the 'git stash' command, which successfully saved the working directory and index state to the master branch. The second line of code has been removed.

```
hp@Annapurna-DESKTOP-B095P65 MINGW64 ~/OneDrive/Documents/git-practice (master)
$ git stash
Saved working directory and index state WIP on master: 860f474 Committed File-1-1

hp@Annapurna-DESKTOP-B095P65 MINGW64 ~/OneDrive/Documents/git-practice (master)
$
```

In the above screenshot what happened is I have run the **git stash** command so the line2 have been disappeared.

I have performed some commands related to git hub

git show – It shows the count of the changes that are made in the files.

git stash list – This command shows the details of the stashed changes.

git stash apply – It will show the changes which are stashed.

```
hp@Annapurna-DESKTOP-B095P65 MINGW64 ~/OneDrive/Documents/git-practice (master)
$ git stash show
File-1 | 1 +
1 file changed, 1 insertion(+)

hp@Annapurna-DESKTOP-B095P65 MINGW64 ~/OneDrive/Documents/git-practice (master)
$ git stash list
stash@{0}: WIP on master: 860f474 Committed File-1-1

hp@Annapurna-DESKTOP-B095P65 MINGW64 ~/OneDrive/Documents/git-practice (master)
$ git stash aply
fatal: subcommand wasn't specified; 'push' can't be assumed due to unexpected token 'aply'

hp@Annapurna-DESKTOP-B095P65 MINGW64 ~/OneDrive/Documents/git-practice (master)
$ git stash apply
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
```

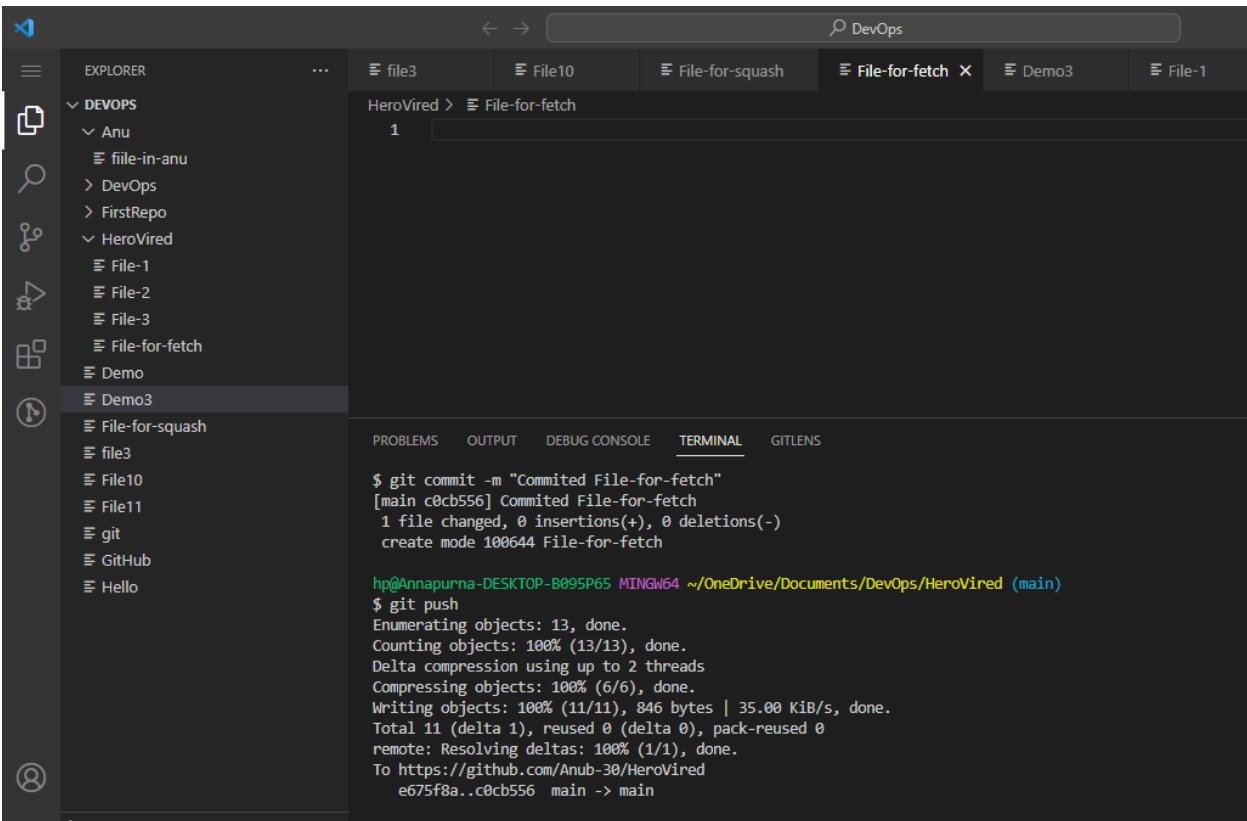
In the above screenshot when we run the command git stash apply it showed the change(line2) which was stashed.

**Q2. By using a sample example of your choice, use the git fetch command and also use the git merge command and describe the whole process through a screenshot with all the commands and their output in git bash.**

## git fetch

git fetch is a command used to download contents from a remote repository.

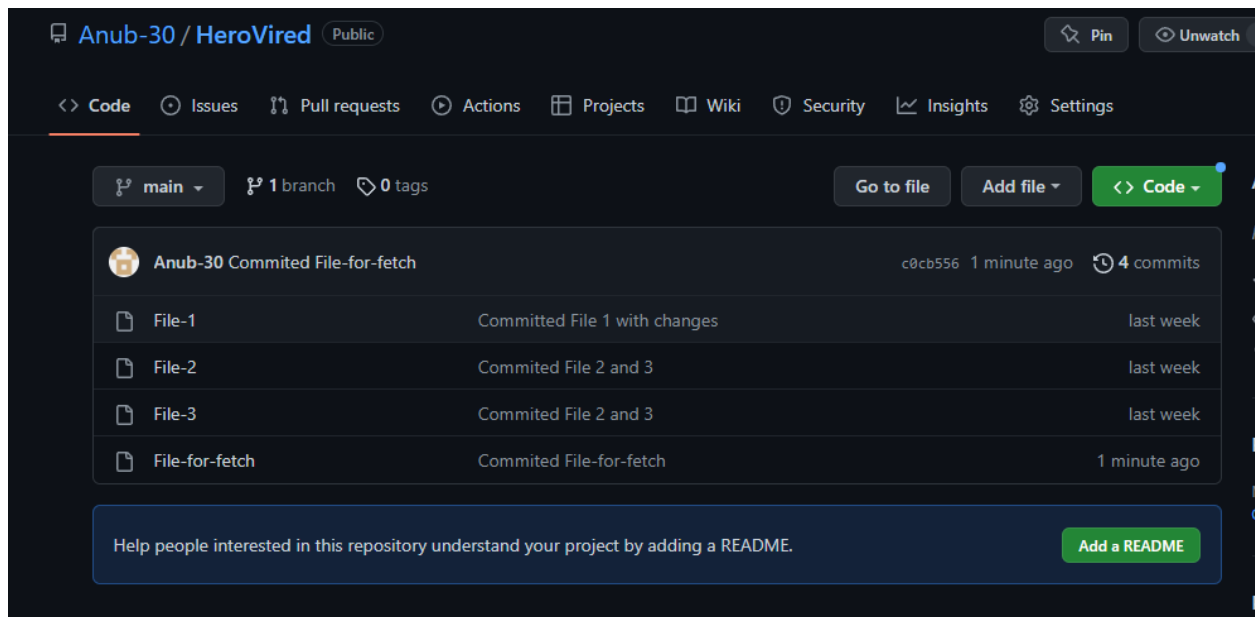
I have a repository in my GitHub called HeroVired. So I have accessed it through my local machine and I have added a file named File-for-fetch and I have pushed it to the repository with the help of **git push** command.

A screenshot of the Visual Studio Code interface. The Explorer panel on the left shows a project structure with folders 'DEVOPS' and 'HeroVired', and files 'File-1', 'File-2', 'File-3', 'File-for-fetch', 'Demo', and 'Demo3'. The 'File-for-fetch' file is selected. The main editor area shows the content of 'File-for-fetch' with the number '1'. The bottom panel shows the 'TERMINAL' tab with the following output:

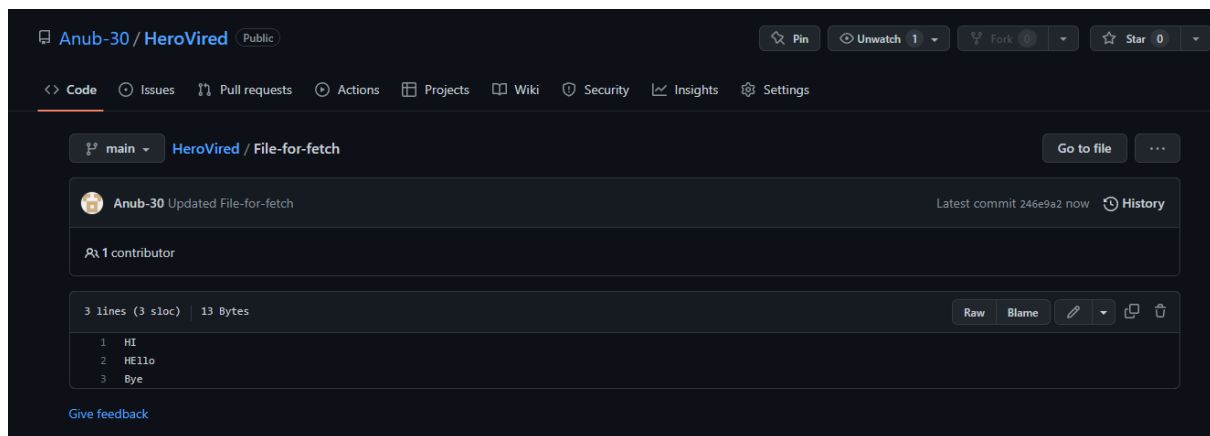
```
$ git commit -m "Committed File-for-fetch"
[main c0cb556] Committed File-for-fetch
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 File-for-fetch

hp@Annapurna-DESKTOP-B095P65 MINGW64 ~/OneDrive/Documents/DevOps/HeroVired (main)
$ git push
Enumerating objects: 13, done.
Counting objects: 100% (13/13), done.
Delta compression using up to 2 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (11/11), 846 bytes | 35.00 KiB/s, done.
Total 11 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/Anub-30/HeroVired
e675f8a..c0cb556 main -> main
```

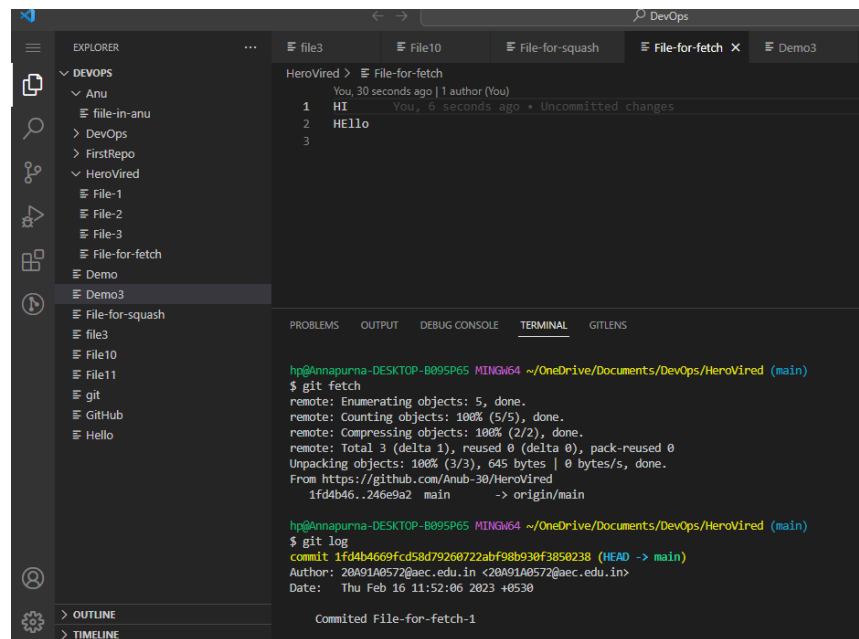
Now I have logged in to my GitHub account and can see the file File-for-fetch as I have pushed it into the repository.



Now I have opened the file and added the line 3 through the edit option and I have committed it in the GitHub.



Now I have performed the **git fetch** command.



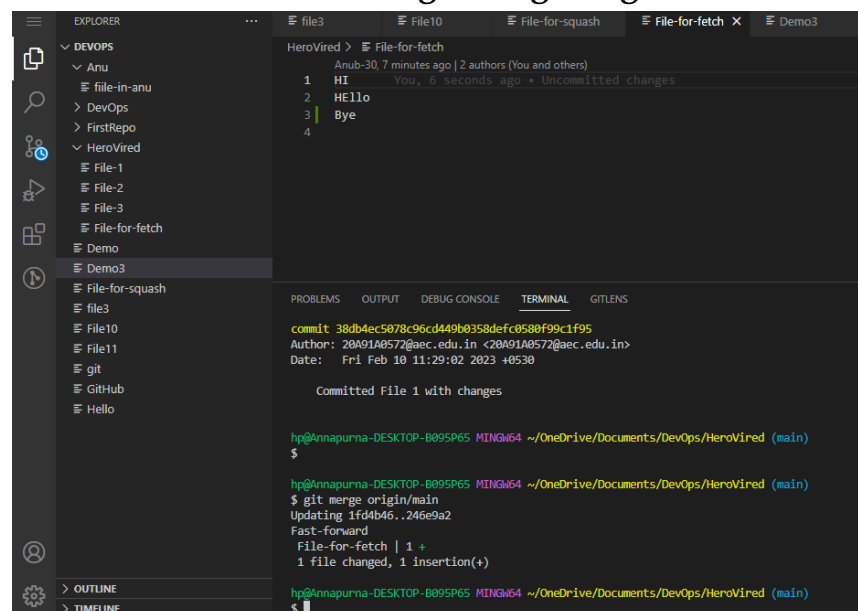
The screenshot shows the VS Code interface with the Explorer sidebar on the left displaying a project structure under 'DEVOPS'. The main editor area is split into two panes. The top pane shows the 'File-for-fetch' file with three lines of text: '1 HI', '2 HELLO', and '3'. The bottom pane is the TERMINAL, showing the output of a 'git fetch' command. The output indicates that the remote repository was updated with new objects and that the local 'main' branch was updated to 'origin/main' with commit '1fd4b46..246e9a2'. Below the fetch output, the 'git log' command is executed, showing a commit by '20A91A8572@aec.edu.in' with the message 'Committed File-for-fetch-1'.

```
hp@Annapurna-DESKTOP-B095P65 MINGW64 ~/OneDrive/Documents/DevOps/HeroVired (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), 645 bytes | 0 bytes/s, done.
From https://github.com/anub-30/HeroVired
 1fd4b46..246e9a2  main    -> origin/main

hp@Annapurna-DESKTOP-B095P65 MINGW64 ~/OneDrive/Documents/DevOps/HeroVired (main)
$ git log
commit 1fd4b4669fcd58d79260722abf98b930f3850238 (HEAD -> main)
Author: 20A91A8572@aec.edu.in <20A91A8572@aec.edu.in>
Date: Thu Feb 16 11:52:06 2023 +0530

    Committed File-for-fetch-1
```

To merge the changes that are made in the remote repository to the local repository we have to use the command **git merge origin/<branch-name>**.



The screenshot shows the VS Code interface with the Explorer sidebar on the left. The main editor area is split into two panes. The top pane shows the 'File-for-fetch' file with four lines of text: '1 HI', '2 HELLO', '3 Bye', and '4'. The bottom pane is the TERMINAL, showing the output of a 'git merge' command. The output indicates that the local 'main' branch was updated with the changes from 'origin/main' (commit '38db4ec5078c96cd449b0358defc0580f99c1f95') and that the local 'main' branch was updated to 'origin/main' with commit '1fd4b46..246e9a2'. Below the merge output, the 'git log' command is executed, showing a commit by '20A91A8572@aec.edu.in' with the message 'Committed File 1 with changes'.

```
commit 38db4ec5078c96cd449b0358defc0580f99c1f95
Author: 20A91A8572@aec.edu.in <20A91A8572@aec.edu.in>
Date: Fri Feb 10 11:29:02 2023 +0530

    Committed File 1 with changes

hp@Annapurna-DESKTOP-B095P65 MINGW64 ~/OneDrive/Documents/DevOps/HeroVired (main)
$
hp@Annapurna-DESKTOP-B095P65 MINGW64 ~/OneDrive/Documents/DevOps/HeroVired (main)
$ git merge origin/main
Updating 1fd4b46..246e9a2
Fast-forward
 File-for-fetch | 1 +
 1 file changed, 1 insertion(+)

hp@Annapurna-DESKTOP-B095P65 MINGW64 ~/OneDrive/Documents/DevOps/HeroVired (main)
$
```

So, now we can see the line3 which we added into the repository in the github.

**Q3. State the difference between git fetch and git pull by doing a practical example in your git bash and attach a screenshot of all the processes.**

### **git fetch**

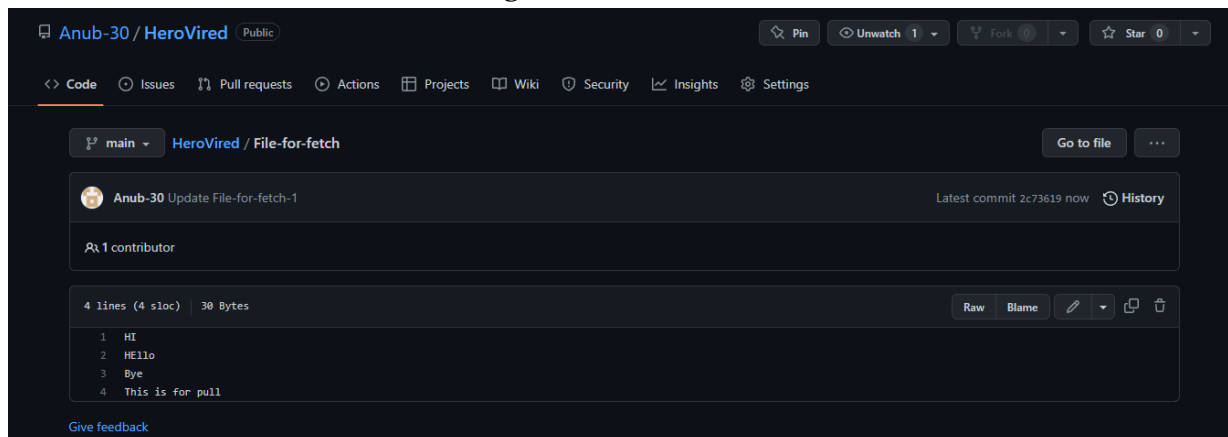
Git Fetch is the command that tells the local repository that there are changes available in the remote repository without bringing the changes into the local repository.

### **git pull**

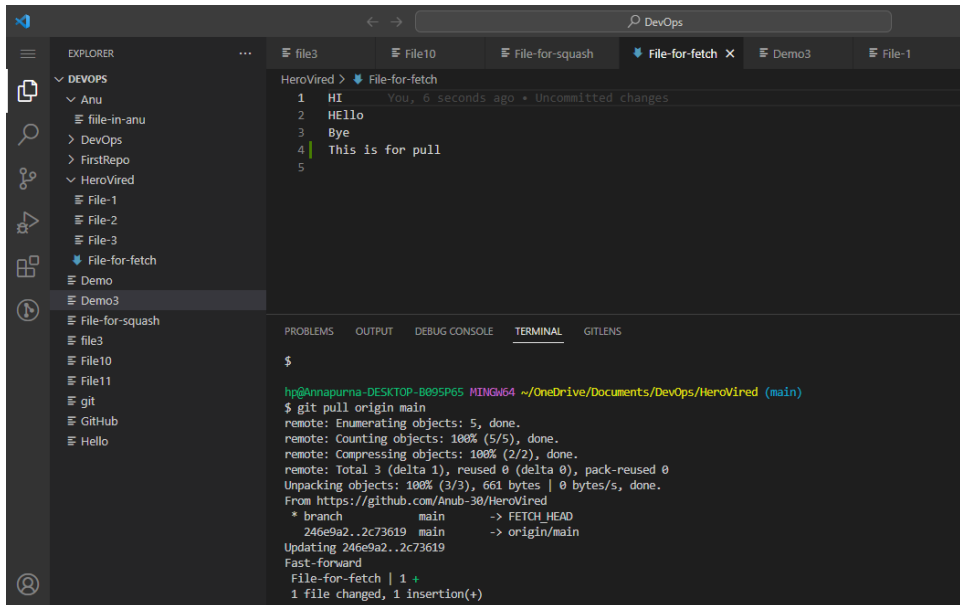
Git Pull on the other hand brings the copy of the remote directory changes into the local repository.

As I have seen the git fetch command in the above question it used to download the changes from the remote repository.

Now for the git pull first I have added line4 in the file File-for-fetch in the github website and committed the changes.



Now if we run the **git pull origin <branch-name>** command it do the operation of the **git fetch** and the **git merge**.



So, here after running the **git pull origin main** command it shows the changes that we made in the File-for-fetch file which we committed in the github website.

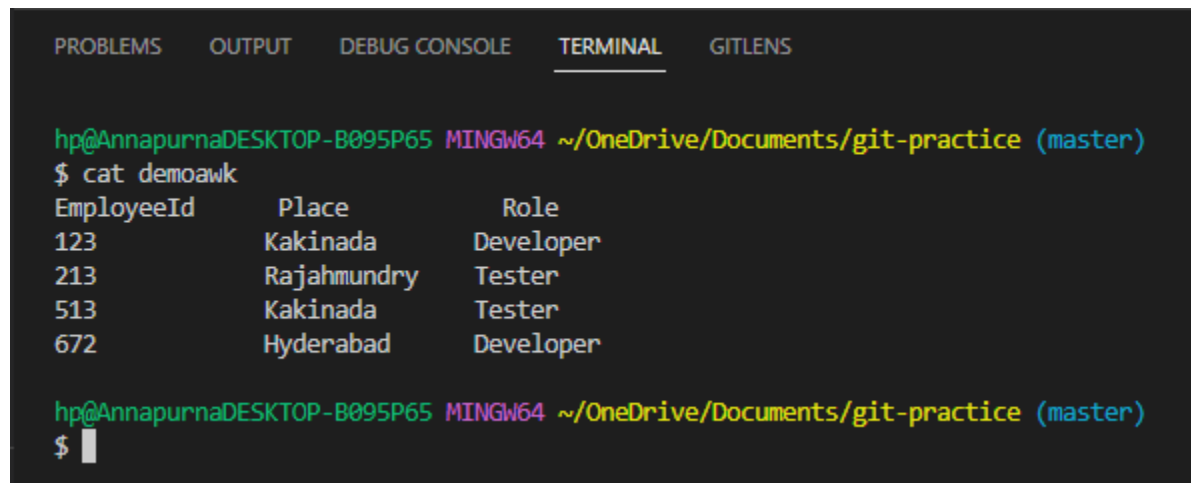


**Q4. Try to find out about the awk command and use it while reading a file created by yourself. Also, make a bash script file and try to find out the prime number from the range 1 to 20.**

**The whole process should be carried out and by using the history command, give the screenshot of all the processes being carried out.**

### **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.



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL GITLENS

hp@AnnapurnaDESKTOP-B095P65 MINGW64 ~/OneDrive/Documents/git-practice (master)
$ cat demoawk
EmployeeId      Place      Role
123             Kakinada   Developer
213             Rajahmundry Tester
513             Kakinada   Tester
672             Hyderabad  Developer

hp@AnnapurnaDESKTOP-B095P65 MINGW64 ~/OneDrive/Documents/git-practice (master)
$
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    GITLENS

513          Kakinada      Tester
672          Hyderabad    Developer

hp@AnnapurnaDESKTOP-B095P65 MINGW64 ~/OneDrive/Documents/git-practice (master)
$ awk '{print}' demoawk
EmployeeId    Place        Role
123           Kakinada     Developer
213           Rajahmundry  Tester
513           Kakinada     Tester
672           Hyderabad  Developer

hp@AnnapurnaDESKTOP-B095P65 MINGW64 ~/OneDrive/Documents/git-practice (master)
$
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    GITLENS

hp@AnnapurnaDESKTOP-B095P65 MINGW64 ~/OneDrive/Documents/git-practice (master)
$ awk '/Developer/ {print}' demoawk
123          Kakinada     Developer
672          Hyderabad    Developer

hp@AnnapurnaDESKTOP-B095P65 MINGW64 ~/OneDrive/Documents/git-practice (master)
$
```

Program of finding prime numbers between 1 to 20 numbers using bash script.

The screenshot shows the Visual Studio Code interface with a file explorer on the left showing 'GIT-PRACTICE' and 'primenumber.sh'. The main editor displays the following bash script:

```
1 #!/bin/bash
2 for((i=2;i<20;))
3 do
4     for((j=i-1;j>=2;))
5     do
6         if [ `expr $i % $j` -ne 0 ] ; then
7             prime=1
8         else
9             prime=0
10            break
11        fi
12        j=`expr $j - 1`
13    done
14    if [ $prime == 1 ] ; then
15        echo $i
16    fi
17    i=`expr $i + 1`
18 done
```

The terminal at the bottom shows the command `$ bash primenumber.sh` being executed. The output displays the numbers 3, 5, 7, 11, 13, 17, and 19. However, there is an error message: `primenumber.sh: line 14: [: ==: unary operator expected`.

**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.**

```
C:\Users\hp>docker run -it ubuntu
Unable to find image 'ubuntu:latest' locally
latest: Pulling from library/ubuntu
677076032cca: Pull complete
Digest: sha256:9a0bdde4188b896a372804be2384015e90e3f84906b750c1a53539b585fbb7f
Status: Downloaded newer image for ubuntu:latest
root@f2225e131dab:/# ls
bin boot dev etc home lib lib32 lib64 libx32 media mnt opt proc root run sbin srv sys tmp usr var
root@f2225e131dab:/#
```