Username and Email:

```
vaishnavi@LAPTOP-12GMAH4Q MINGW64 ~ (master)
$ git config user.name
vaishnavipula

vaishnavi@LAPTOP-12GMAH4Q MINGW64 ~ (master)
$ git config user.email
vaishnavipula26@gmail.com

vaishnavi@LAPTOP-12GMAH4Q MINGW64 ~ (master)
$
```

Question-1:

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:

Stash is a Git command to locally store your recent changes in a separate area so you can fetch those changes later. Stash uses "stack" data structure. After taking a snapshot of your local files, it resets the state of your workspace to the previous commit state. You can save multiple stashes on your local computer and you can apply back any of the stashes at a later stage.

Using "git init" I have initialized a repository and created files file1 and file2 in the directory. I made changes to both files and file1 is added to staged area and file2 is just modified.

```
vaishnavi@LAPTOP-12GMAH4Q MINGW64 ~/Git-Stash (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   file1

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   file2
```

git stash: git stash is used to stash changes.

git stash list: It is used to show the stash list in the first in last out manner (Stack data structure).

```
vaishnavi@LAPTOP-12GMAH4Q MINGW64 ~/Git-Stash (master)
$ git stash
warning: LF will be replaced by CRLF in file2.
The file will have its original line endings in your working directory
Saved working directory and index state WIP on master: 1be808d 3 files added

vaishnavi@LAPTOP-12GMAH4Q MINGW64 ~/Git-Stash (master)
$ git stash list
stash@{0}: WIP on master: 1be808d 3 files added
```

git stash push: It is used to push changes to stash. Git stash applies default push .In git stash push we have to enter message.

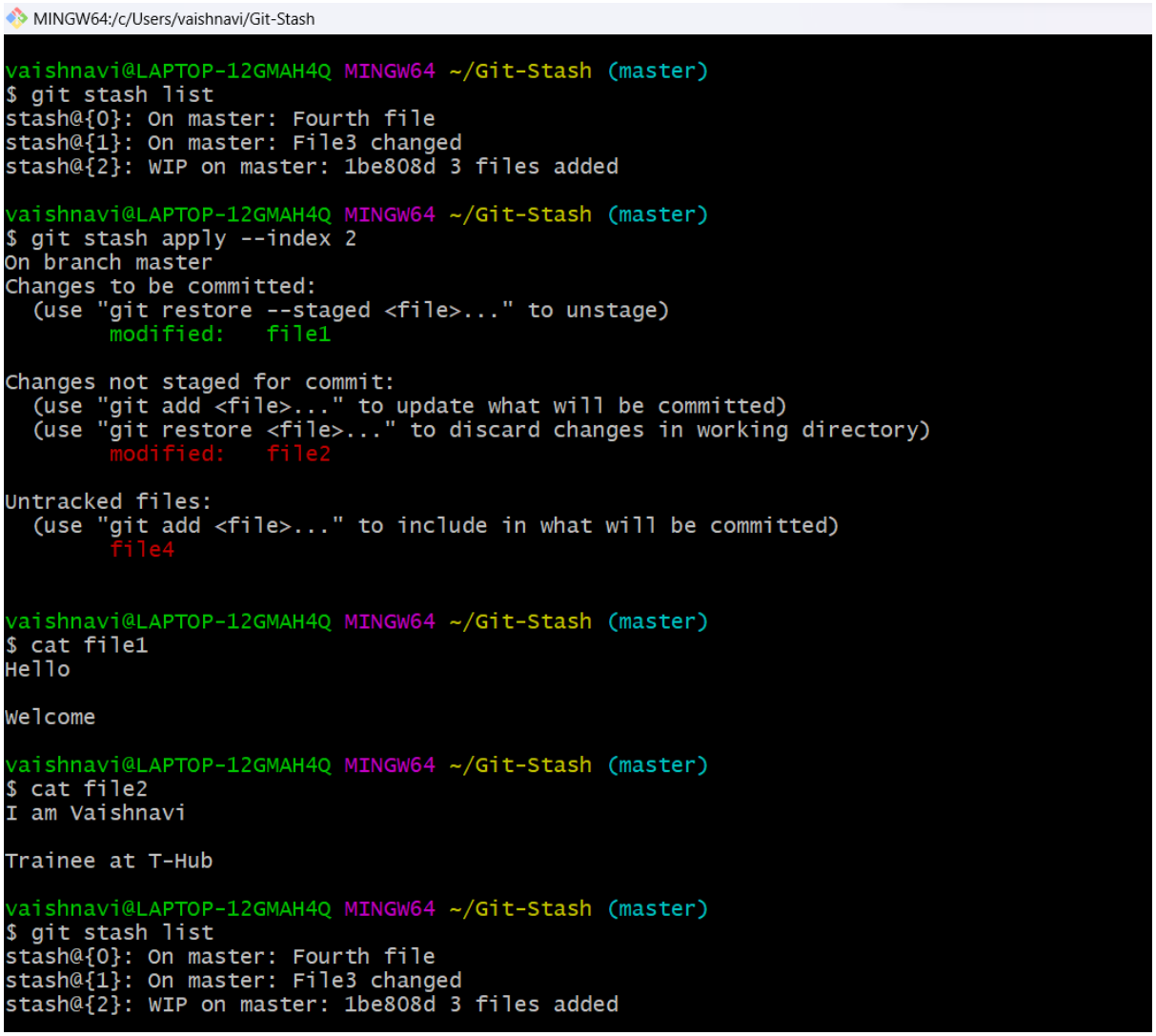
Modified file3 and pushed to stash.

```
vaishnavi@LAPTOP-12GMAH4Q MINGW64 ~/Git-Stash (master)
$ cat >> file3
This is file3

[3]+  Stopped                 cat >> file3

vaishnavi@LAPTOP-12GMAH4Q MINGW64 ~/Git-Stash (master)
$ cat file3
This is file3

vaishnavi@LAPTOP-12GMAH4Q MINGW64 ~/Git-Stash (master)
$ git stash push -m "File3 changed"
warning: LF will be replaced by CRLF in file3.
The file will have its original line endings in your working directory
Saved working directory and index state On master: File3 changed

vaishnavi@LAPTOP-12GMAH4Q MINGW64 ~/Git-Stash (master)
$ git stash list
stash@{0}: On master: File3 changed
stash@{1}: WIP on master: 1be808d 3 files added
```

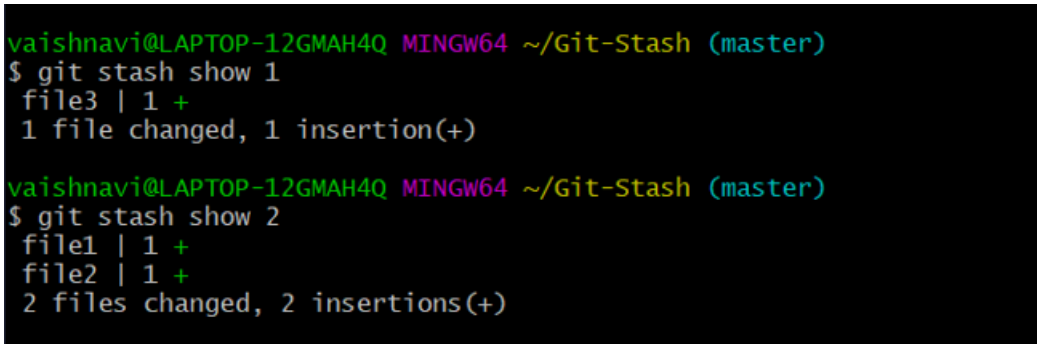
Modified file1 and file4 and changes pushed to stash

```
MINGW64:/c/Users/vaishnavi/Git-Stash

vaishnavi@LAPTOP-12GMAH4Q MINGW64 ~/Git-Stash (master)
$ cat >> file1
This is Git Stash

[4]+  Stopped                 cat >> file1

vaishnavi@LAPTOP-12GMAH4Q MINGW64 ~/Git-Stash (master)
$ git stash push -m "File1 changed"
No local changes to save

vaishnavi@LAPTOP-12GMAH4Q MINGW64 ~/Git-Stash (master)
$ cat file1
Hello

This is Git Stash

vaishnavi@LAPTOP-12GMAH4Q MINGW64 ~/Git-Stash (master)
$ cat > file4
This is fourth file

[5]+  Stopped                 cat > file4

vaishnavi@LAPTOP-12GMAH4Q MINGW64 ~/Git-Stash (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   file1

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        file4

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

vaishnavi@LAPTOP-12GMAH4Q MINGW64 ~/Git-Stash (master)
$ git stash push -m "Fourth file"
warning: LF will be replaced by CRLF in file1.
The file will have its original line endings in your working directory
Saved working directory and index state On master: Fourth file
```

git stash apply: It applies the changes back to file in working directory. We have to mention index of stash in list.



git stash show: It is used to show the changes made.It indicates files and number of insertions.



git stash pop: It is used to remove specified stash from the list. Changes made at particular stash will be removed if not applied before pop.

```
vaishnavi@LAPTOP-12GMAH4Q MINGW64 ~/Git-Stash (master)
$ git stash pop 1
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   file1

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   file2
        modified:   file3

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        file4

Dropped refs/stash@{1} (6376c546fd140d264697950d38db800ccb7c863b)

vaishnavi@LAPTOP-12GMAH4Q MINGW64 ~/Git-Stash (master)
$ cat file3
This is file3

vaishnavi@LAPTOP-12GMAH4Q MINGW64 ~/Git-Stash (master)
$ git stash list
stash@{0}: On master: Fourth file
stash@{1}: WIP on master: 1be808d 3 files added
```

git stash clear: It clears entire stash list.

```
vaishnavi@LAPTOP-12GMAH4Q MINGW64 ~/Git-Stash (master)
$ git stash list
stash@{0}: On master: Fourth file
stash@{1}: WIP on master: 1be808d 3 files added

vaishnavi@LAPTOP-12GMAH4Q MINGW64 ~/Git-Stash (master)
$ git stash clear

vaishnavi@LAPTOP-12GMAH4Q MINGW64 ~/Git-Stash (master)
$ git stash list

vaishnavi@LAPTOP-12GMAH4Q MINGW64 ~/Git-Stash (master)
```

Question-2:

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: It fetches all the changes from the remote repository and stores it in a separate branch in your local repository. You can reflect those changes in your corresponding branches by merging.

git fetch URL

```
vaishnavi@LAPTOP-12GMAH4Q MINGW64 ~ (master)
$ mkdir fetch

vaishnavi@LAPTOP-12GMAH4Q MINGW64 ~ (master)
$ cd fetch

vaishnavi@LAPTOP-12GMAH4Q MINGW64 ~/fetch (master)
$ git fetch https://github.com/Vaishnavipula/FILE.git
remote: Enumerating objects: 16, done.
remote: Counting objects: 100% (16/16), done.
remote: Compressing objects: 100% (9/9), done.
remote: Total 16 (delta 0), reused 16 (delta 0), pack-reused 0
Unpacking objects: 100% (16/16), 1.34 KiB | 13.00 KiB/s, done.
From https://github.com/Vaishnavipula/FILE
 * branch            HEAD       -> FETCH_HEAD

vaishnavi@LAPTOP-12GMAH4Q MINGW64 ~/fetch (master)
$
```

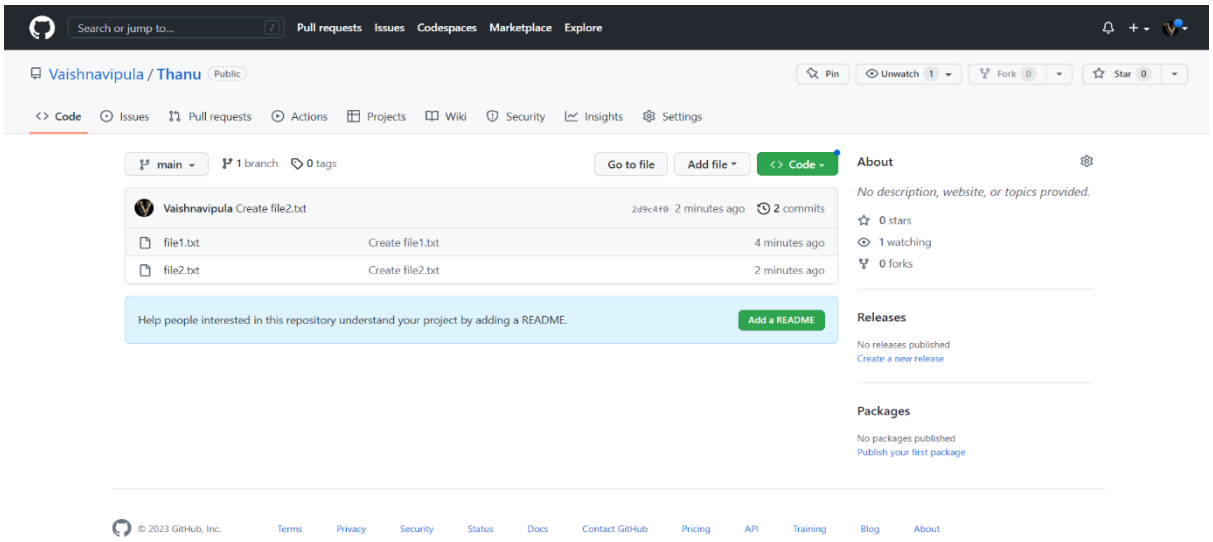
git fetch URL branch_name

```
vaishnavi@LAPTOP-12GMAH4Q MINGW64 ~/fetch (master)
$ git fetch https://github.com/Vaishnavipula/FILE.git test
remote: Enumerating objects: 19, done.
remote: Counting objects: 100% (19/19), done.
remote: Compressing objects: 100% (11/11), done.
remote: Total 19 (delta 1), reused 15 (delta 0), pack-reused 0
Unpacking objects: 100% (19/19), 1.94 KiB | 55.00 KiB/s, done.
From https://github.com/Vaishnavipula/FILE
 * branch            test       -> FETCH_HEAD

vaishnavi@LAPTOP-12GMAH4Q MINGW64 ~/fetch (master)
$
```

git fetch --all

```
vaishnavi@LAPTOP-12GMAH4Q MINGW64 ~/fetch-2 (master)
$ git remote add origin https://github.com/Vaishnavipula/FILE.git

vaishnavi@LAPTOP-12GMAH4Q MINGW64 ~/fetch-2 (master)
$ git fetch --all
Fetching origin
remote: Enumerating objects: 22, done.
remote: Counting objects: 100% (22/22), done.
remote: Compressing objects: 100% (13/13), done.
remote: Total 22 (delta 2), reused 15 (delta 0), pack-reused 0
Unpacking objects: 100% (22/22), 2.54 KiB | 39.00 KiB/s, done.
From https://github.com/Vaishnavipula/FILE
 * [new branch]      master     -> origin/master
 * [new branch]      test       -> origin/test
 * [new branch]      test2      -> origin/test2

vaishnavi@LAPTOP-12GMAH4Q MINGW64 ~/fetch-2 (master)
$
```

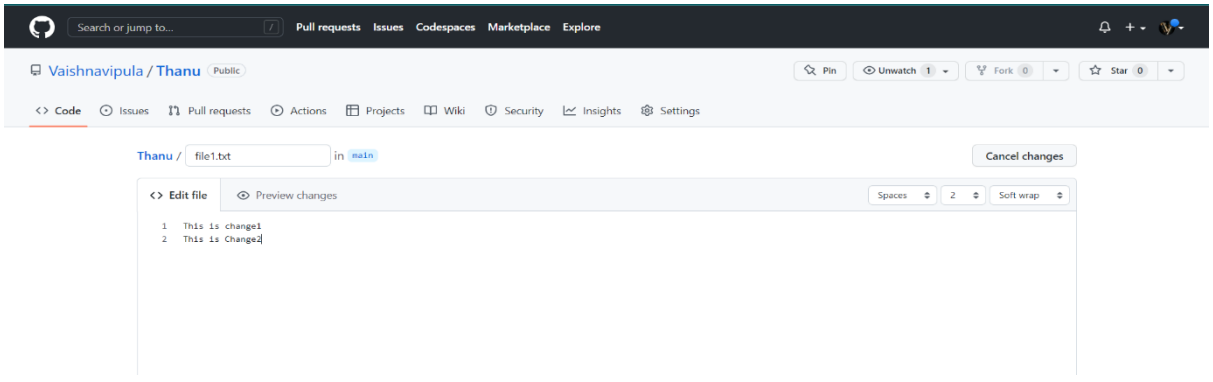
Let us create a repository in Github and add files in it.



Let us clone the repository from remote to local using "git clone URL"

```
vaishnavi@LAPTOP-12GMAH4Q MINGW64 ~/Fetch_dir (master)
$ git clone https://github.com/Vaishnavipula/Thanu.git
Cloning into 'Thanu'...
remote: Enumerating objects: 6, done.
remote: Counting objects: 100% (6/6), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 6 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (6/6), done.

vaishnavi@LAPTOP-12GMAH4Q MINGW64 ~/Fetch_dir (master)
$ ls
Thanu/

vaishnavi@LAPTOP-12GMAH4Q MINGW64 ~/Fetch_dir (master)
$ cd Thanu

vaishnavi@LAPTOP-12GMAH4Q MINGW64 ~/Fetch_dir/Thanu (main)
$ ls
file1.txt  file2.txt

vaishnavi@LAPTOP-12GMAH4Q MINGW64 ~/Fetch_dir/Thanu (main)
$ cat file1.txt
This is change1
```

Now make a change in any file in remote repository



Use git fetch: When we use fetch ,the changes made in remote file won't reflect in local files .We just get the details of changes made.

```
vaishnavi@LAPTOP-12GMAH4Q MINGW64 ~/Fetch_dir/Thanu (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 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 673 bytes | 26.00 KiB/s, done.
From https://github.com/Vaishnavipula/Thanu
   2d9c4f0..6fe3986  main       -> origin/main

vaishnavi@LAPTOP-12GMAH4Q MINGW64 ~/Fetch_dir/Thanu (main)
$ cat file1.txt
This is change1
```

Use git merge: After using git merge the changes made in remote repository files will be reflected in local files.

```
vaishnavi@LAPTOP-12GMAH4Q MINGW64 ~/Fetch_dir/Thanu (main)
$ git merge
Updating 2d9c4f0..6fe3986
Fast-forward
 file1.txt | 1 +
 1 file changed, 1 insertion(+)

vaishnavi@LAPTOP-12GMAH4Q MINGW64 ~/Fetch_dir/Thanu (main)
$ cat file1.txt
This is change1
This is Change2

vaishnavi@LAPTOP-12GMAH4Q MINGW64 ~/Fetch_dir/Thanu (main)
$
```

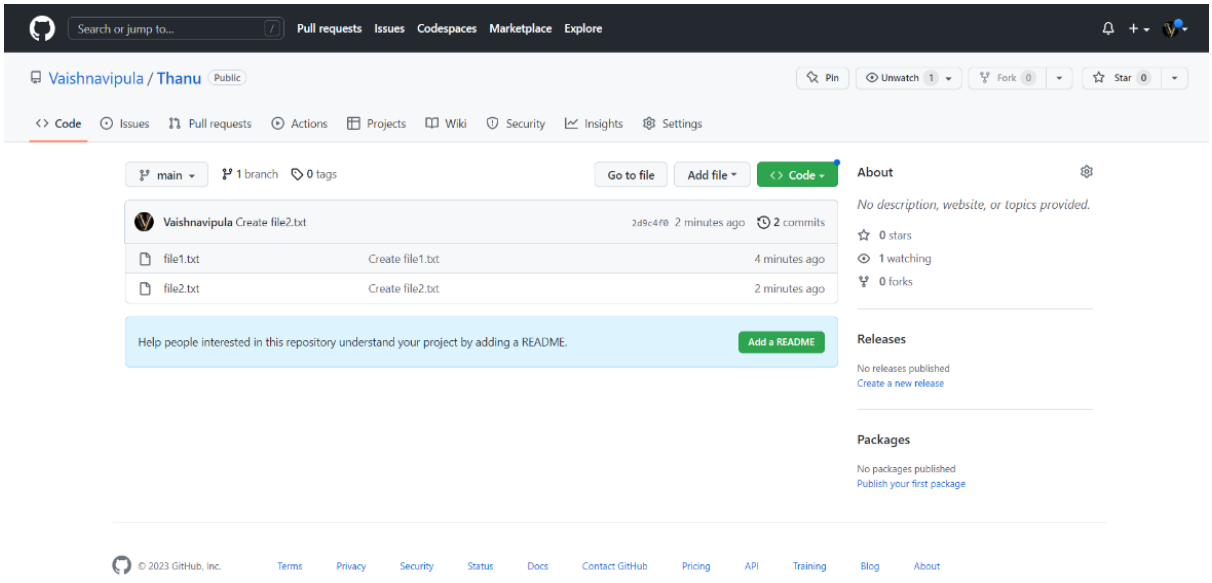
Question-3:

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: It fetches the changes made in remote repository to local repository.When we use fetch there is no file transfer.

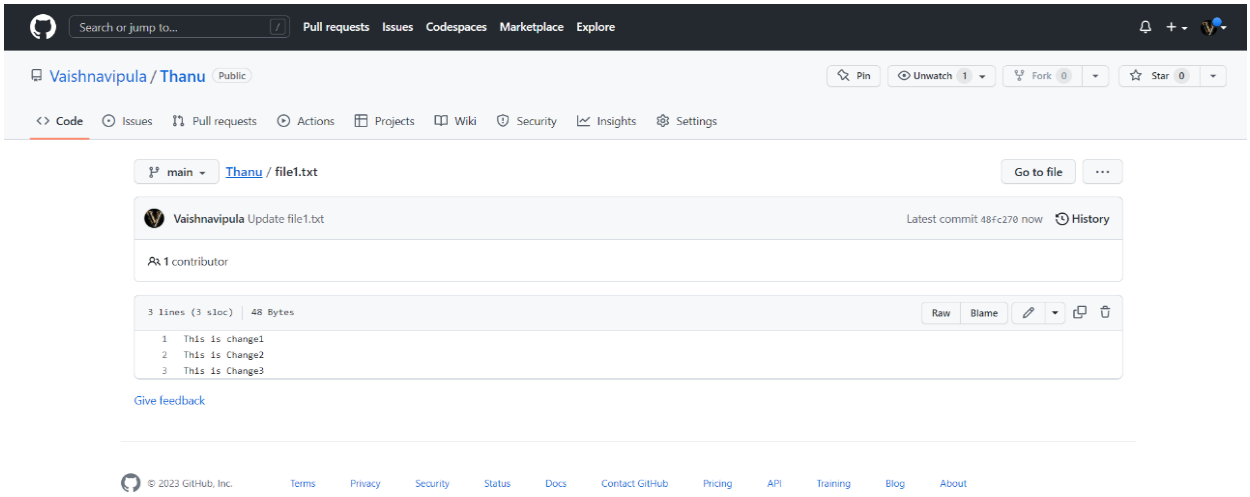
Git Pull: It Pulls the changes made in remote repository to local repository.When we apply Pull , file transfer takes place from remote to local.

We have 2 files in repository named "Thanu".

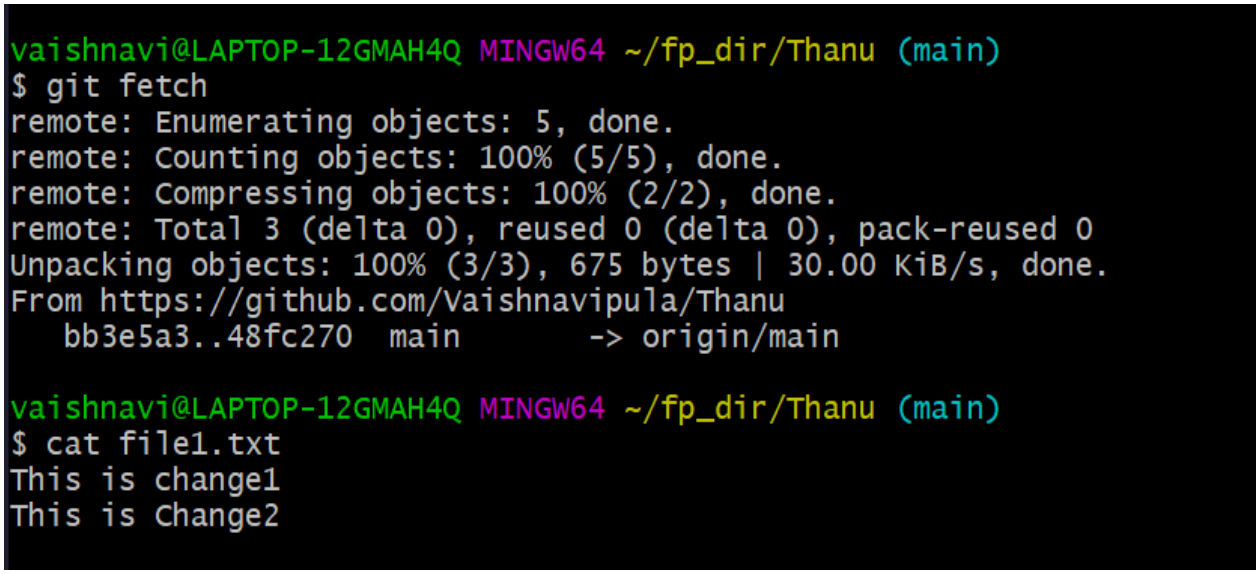


After cloning we will have the files in local repository.

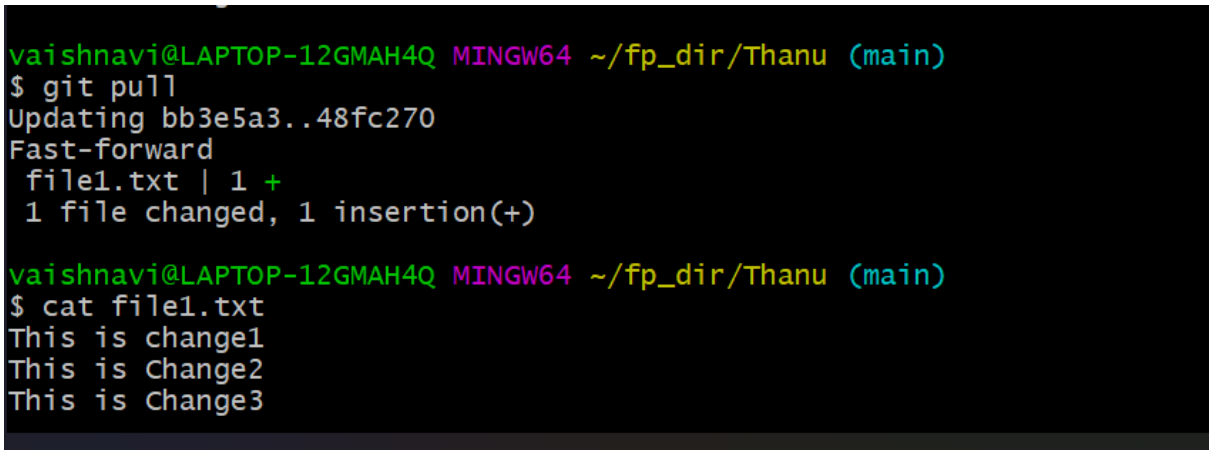
Now make changes to files in Remote Repository.



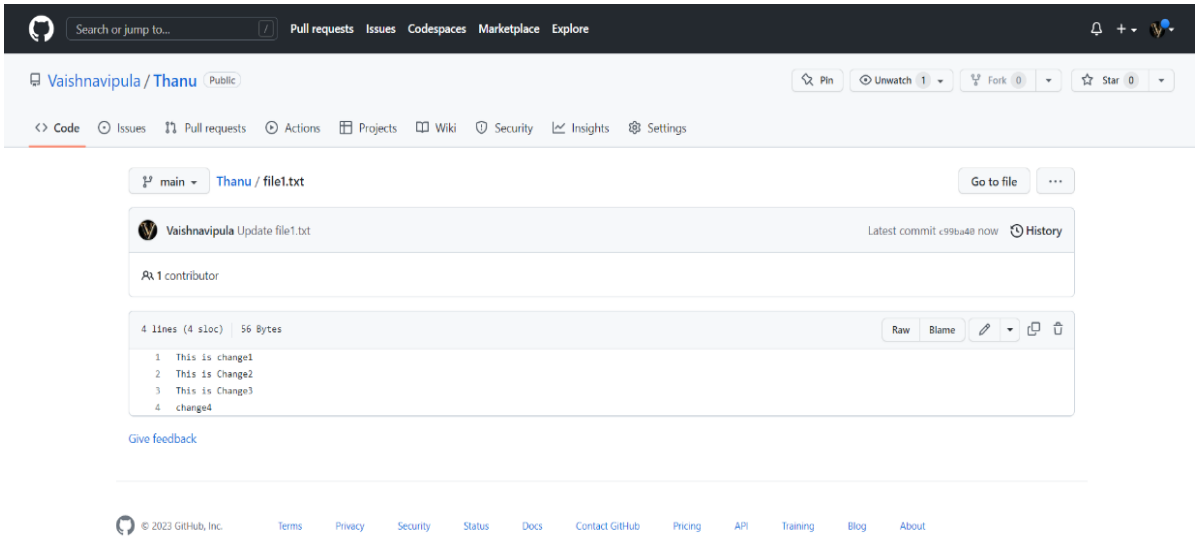
Use git fetch in git bash.



After using git pull , changes made in Remote Repository will be reflected in Local Repository.



git pull: It will do the work done by both Fetch and Merge.




```
vaishnavi@LAPTOP-12GMAH4Q MINGW64 ~/fp_dir/Thanu (main)
$ git pull
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 680 bytes | 37.00 KiB/s, done.
From https://github.com/Vaishnavipula/Thanu
   48fc270..c99ba40  main       -> origin/main
Updating 48fc270..c99ba40
Fast-forward
 file1.txt | 1 +
 1 file changed, 1 insertion(+)

vaishnavi@LAPTOP-12GMAH4Q MINGW64 ~/fp_dir/Thanu (main)
$ cat file1.txt
This is change1
This is Change2
This is Change3
change4
```

Question-4:

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 mostly used for pattern scanning and processing. It searches one or more files to see if they contain lines that matches with the specified patterns and then perform the associated actions.

Let us create a file and apply Awk command. The following syntaxes are used to get required columns.

```
vaishnavi@LAPTOP-12GMAH4Q MINGW64 ~ (master)
$ cd AWK_dir

vaishnavi@LAPTOP-12GMAH4Q MINGW64 ~/AWK_dir (master)
$ cat > file1
Vaishu  552
Thanu   502
Dhanu   520
Anvi    542
Deepu   504

[1]+  Stopped                 cat > file1

vaishnavi@LAPTOP-12GMAH4Q MINGW64 ~/AWK_dir (master)
$ cat file1
Vaishu  552
Thanu   502
Dhanu   520
Anvi    542
Deepu   504

vaishnavi@LAPTOP-12GMAH4Q MINGW64 ~/AWK_dir (master)
$ awk '{print $0}' file1
Vaishu  552
Thanu   502
Dhanu   520
Anvi    542
Deepu   504

vaishnavi@LAPTOP-12GMAH4Q MINGW64 ~/AWK_dir (master)
$ awk '{print $1}' file1
Vaishu
Thanu
Dhanu
Anvi
Deepu

vaishnavi@LAPTOP-12GMAH4Q MINGW64 ~/AWK_dir (master)
$ awk '{print $1,$2}' file1
Vaishu 552
Thanu 502
Dhanu 520
Anvi 542
Deepu 504
```

The following awk commands are used to print serial number along with the required column and prints last column.

```
vaishnavi@LAPTOP-12GMAH4Q MINGW64 ~/AWK_dir (master)
$ awk '{print NR,$1}' file1
1 Vaishu
2 Thanu
3 Dhanu
4 Anvi
5 Deepu

vaishnavi@LAPTOP-12GMAH4Q MINGW64 ~/AWK_dir (master)
$ awk '{print $NF}' file1
552
502
520
542
504
```

The following commands are used to print specified line in required column.

```
vaishnavi@LAPTOP-12GMAH4Q MINGW64 ~/AWK_dir (master)
$ awk '{print $1}' file1 | head -1
Vaishu

vaishnavi@LAPTOP-12GMAH4Q MINGW64 ~/AWK_dir (master)
$ awk '{print $1}' file1 | head -2
Vaishu
Thanu
```

The following commands are used to print the pattern matching lines.

^D prints lines that start with ‘D’ and 0$ prints the lines that end with ‘0’ .

```
vaishnavi@LAPTOP-12GMAH4Q MINGW64 ~/AWK_dir (master)
$ awk '/^D/' file1
Dhanu   520
Deepu   504

vaishnavi@LAPTOP-12GMAH4Q MINGW64 ~/AWK_dir (master)
$ awk '/0$/' file1
Dhanu   520
```

Using Bash Script:

Create a file using vi editor with ".sh" extension and write bash script for printing the prime numbers from the range 1 to 20.

MINGW64:/c/Users/vaishnavi/AWK_dir

```
#!/bin/bash

prime_1=1
n=20
echo " Prime number between 1 to $n is:"
echo "2"
for((i=3;i<=n;))
do
  for((j=i-1;j>=2;))
  do
    if [  `expr $i % $j` -ne 0 ] ; then
      prime_1=1
    else
      prime_1=0
      break
    fi
    j=`expr $j - 1`
  done
  if [ $prime_1 -eq 1 ] ; then
    echo $i
  fi
  i=`expr $i + 1`
done
~
~
```

Use "./filename.sh" to run bash script file.

```
vaishnavi@LAPTOP-12GMAH4Q MINGW64 ~/AWK_dir (master)
$  vi file2.sh

vaishnavi@LAPTOP-12GMAH4Q MINGW64 ~/AWK_dir (master)
$ ./file2.sh
 Prime number between 1 to 20 is:
2
3
5
7
11
13
17
19
```

History command: History command is used to show the list of commands used till now.

```
vaishnavi@LAPTOP-12GMAH4Q MINGW64 ~/AWK_dir (master)
$  history 15
  497  cd AWK_dir
  498  cat > file1
  499  cat file1
  500  awk '{print $0}' file1
  501  awk '{print $1}' file1
  502  awk '{print $1,$2}' file1
  503  awk '{print NR,$1}' file1
  504  awk '{print $NF}' file1
  505  awk '{print $1}' file1 | head -1
  506  awk '{print $1}' file1 | head -2
  507  awk '/^D/' file1
  508  awk '/O$/' file1
  509   vi file2.sh
  510  ./file2.sh
  511   history 15
```

Question-5:

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.

In Docker to run the Ubuntu operating system first we need to pull it from the Docker Hub and then run it by creating a container.

Docker pull: If image is already available then pull updates the image ,otherwise it downloads .

```
C:\Users\vaishnavi>docker pull ubuntu
Using default tag: latest
latest: Pulling from library/ubuntu
Digest: sha256:9a0bdde4188b896a372804be2384015e90e3f84906b750c1a53539b585fbbe7f
Status: Image is up to date for ubuntu:latest
docker.io/library/ubuntu:latest
```

Docker Create: Used to create Containers.

```
C:\Users\vaishnavi>docker create ubuntu
4f72738e2527d0264f39baf419fa98de8aa085adb96f81fa469ffab6f3550c39

C:\Users\vaishnavi>docker ps -all
CONTAINER ID   IMAGE     COMMAND       CREATED         STATUS    PORTS     NAMES
4f72738e2527   ubuntu    "/bin/bash"   6 seconds ago   Created             vigilant_roentgen
```

Docker Run: It is used to run the image.To run Ubuntu in interactive mode we have to use " docker run -it ubuntu " .To exit the process use exit command in interactive mode of ubuntu.

```
C:\Users\vaishnavi>docker run -it ubuntu
root@4936afe1dd5e:/# ls
bin  boot  dev  etc  home  lib  lib32  lib64  libx32  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
root@4936afe1dd5e:/# whoami
root
root@4936afe1dd5e:/# exit
exit

C:\Users\vaishnavi>docker ps -all
CONTAINER ID   IMAGE     COMMAND       CREATED         STATUS                     PORTS     NAMES
4936afe1dd5e   ubuntu    "/bin/bash"   2 minutes ago   Exited (0) 4 seconds ago             awesome_cerf

C:\Users\vaishnavi>
```