SOURCE CODE MANAGEMENT

STUDENT ACTIVITY REPORT

KHUSHI SUNDIP MHAMANE
B.TECH (CSE), BATCH-3
A86605224127

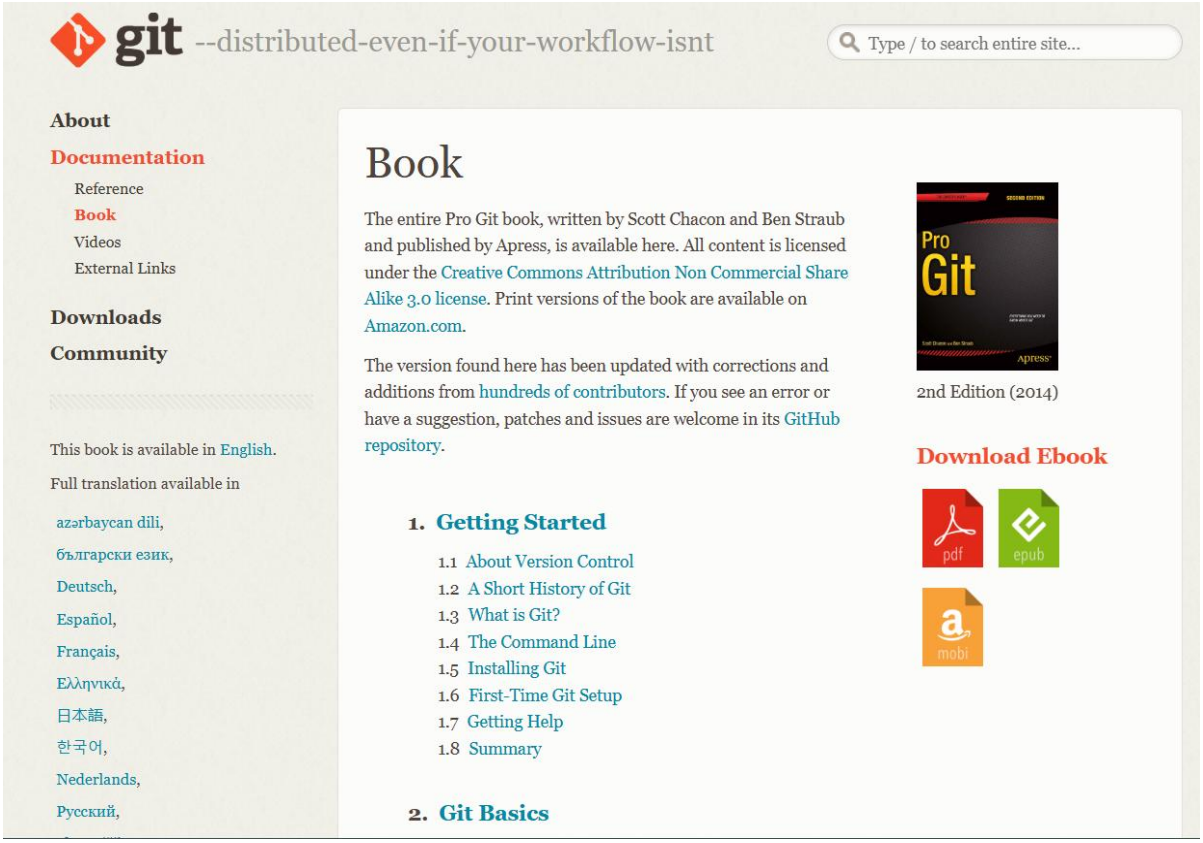
AMITY UNIVERSITY BANGALORE

INDEX

ACTIVITY-1

PART 1 : INSTALLING GIT

- Download the 'PRO GIT' book pdf file, and use the link in the book to go download git from the web.



Installing on Windows

There are also a few ways to install Git on Windows. The most official build is available for download on the Git website. Just go to https://git-scm.com/download/win and the download will start automatically. Note that this is a project called Git for Windows, which is separate from Git itself; for more information on it, go to https://gitforwindows.org.

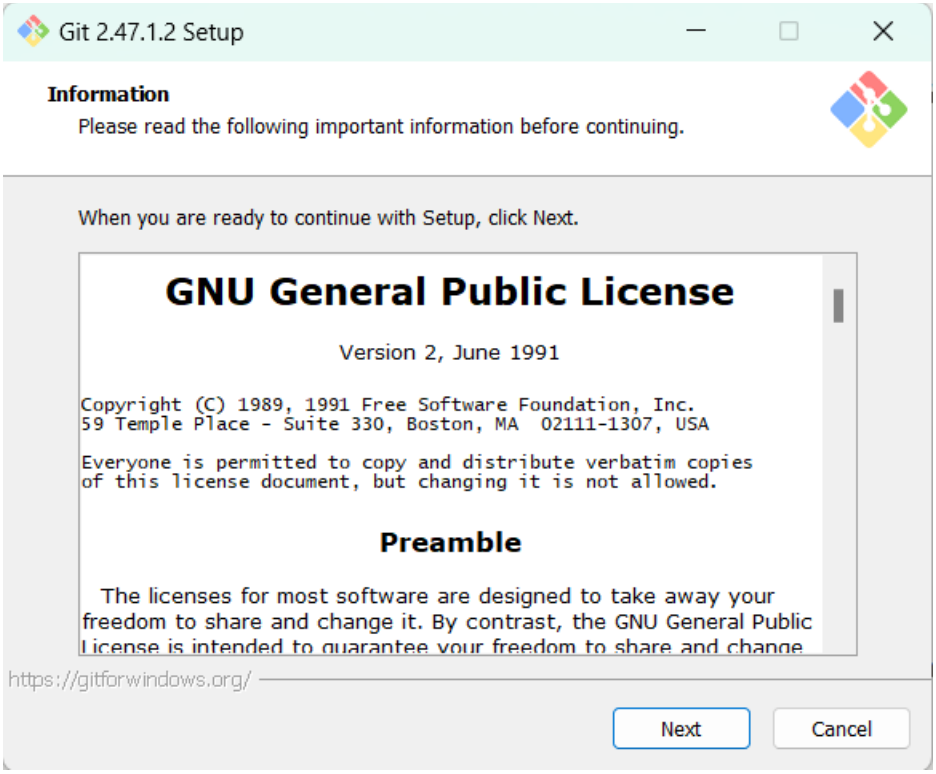
To get an automated installation you can use the Git Chocolatey package. Note that the Chocolatey package is community maintained.

19

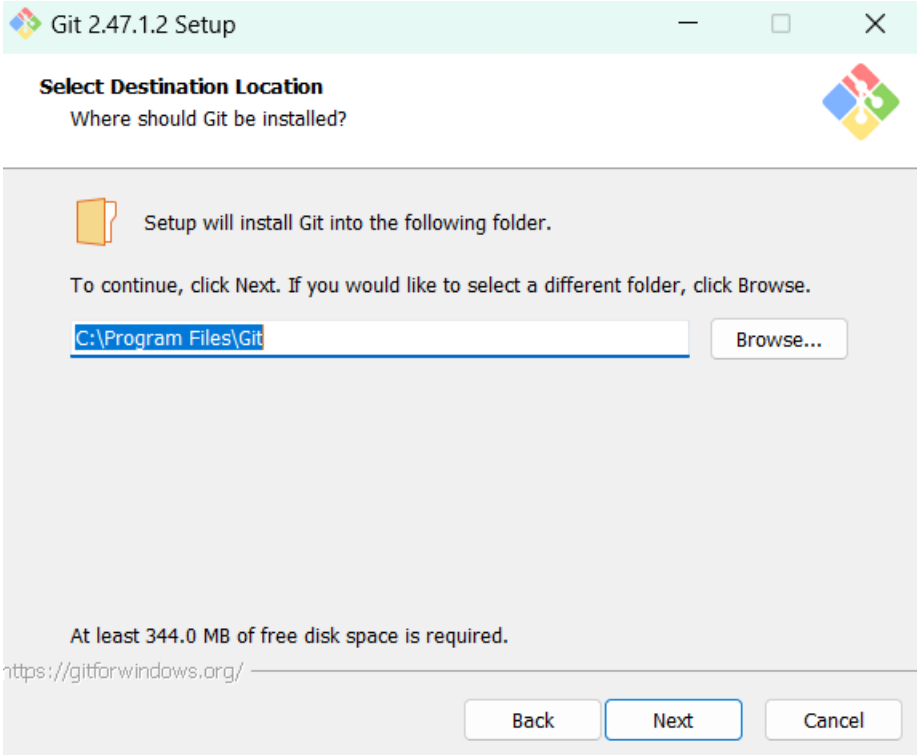
- Select and choose between 32 or 64 bit Git. Here we are going with 64 bit. Open the file after downloading.

#note that 64 bit processes faster than 32 bit

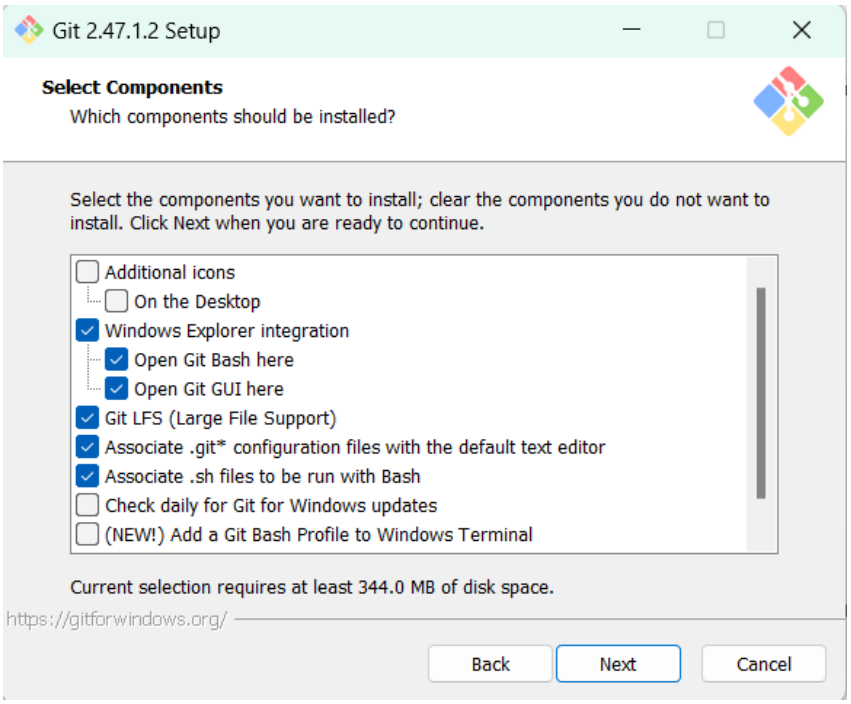
- A screen displaying the license appears. Here the GNU license shows that this is open source, and gives people the freedom to change its source code and distribute it.



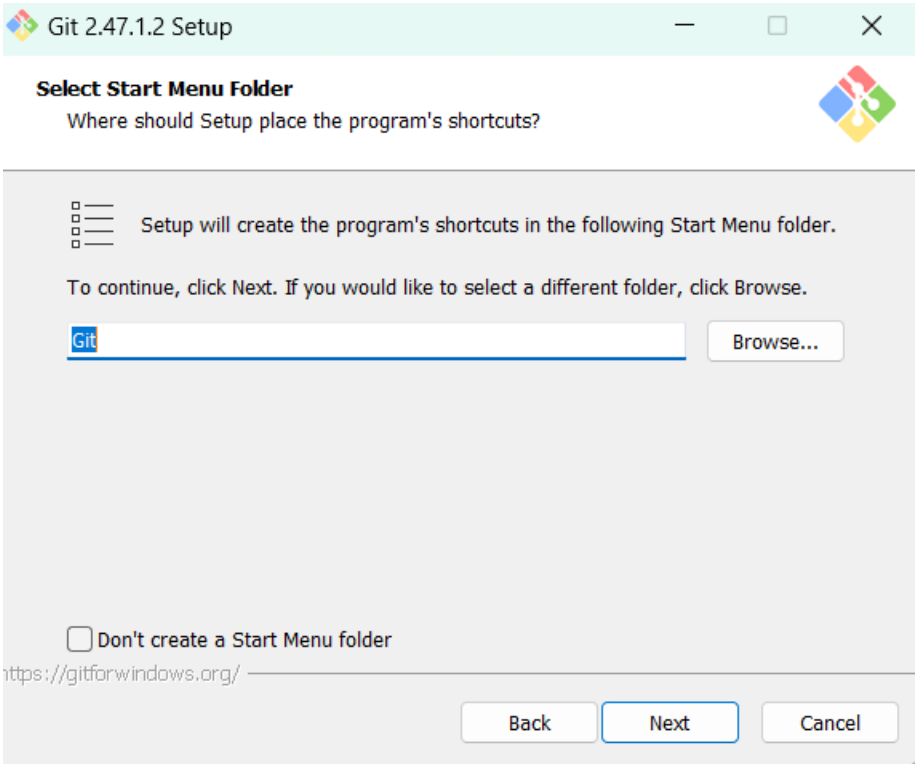
- Select the installation location of your choice.



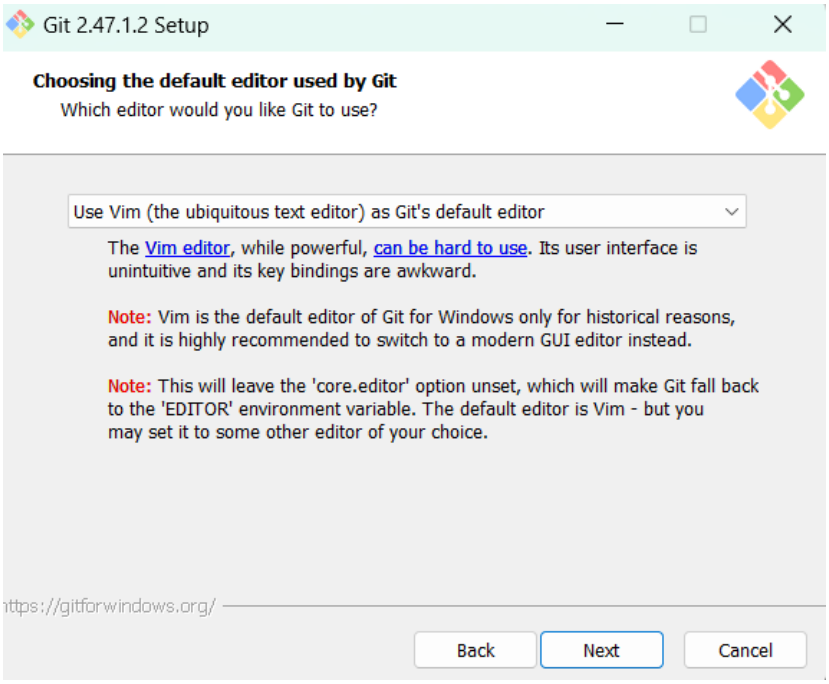
- Here we select the downloadable contents. Git comes with its own text editor.



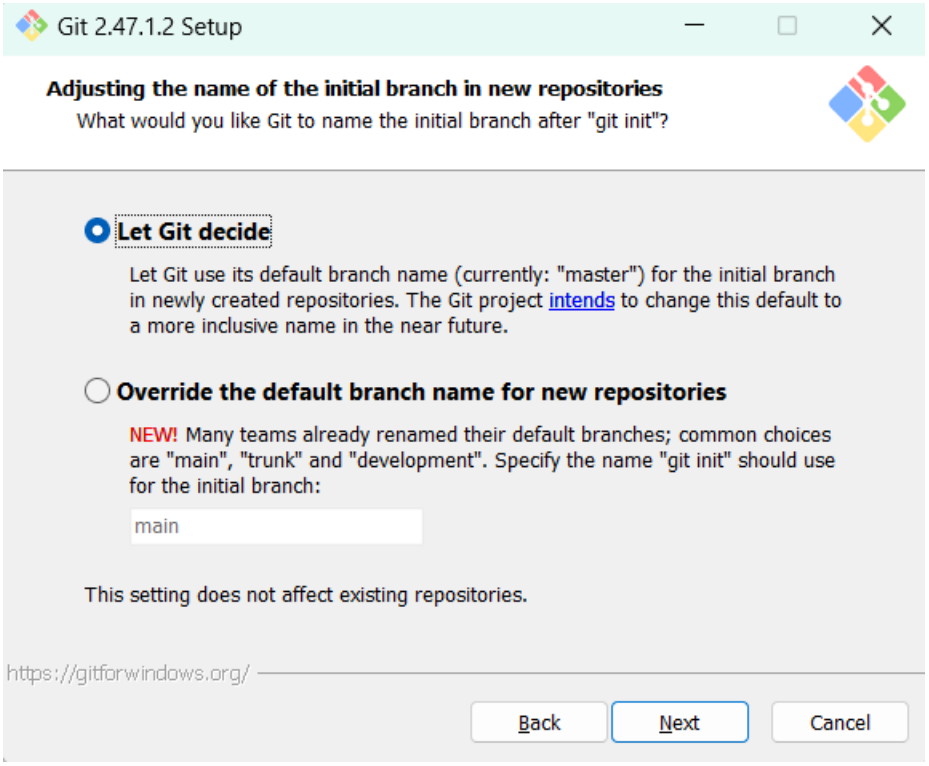
- This creates a folder in the start menu to access the programs easily. It'll be stored in the file called 'Git'.



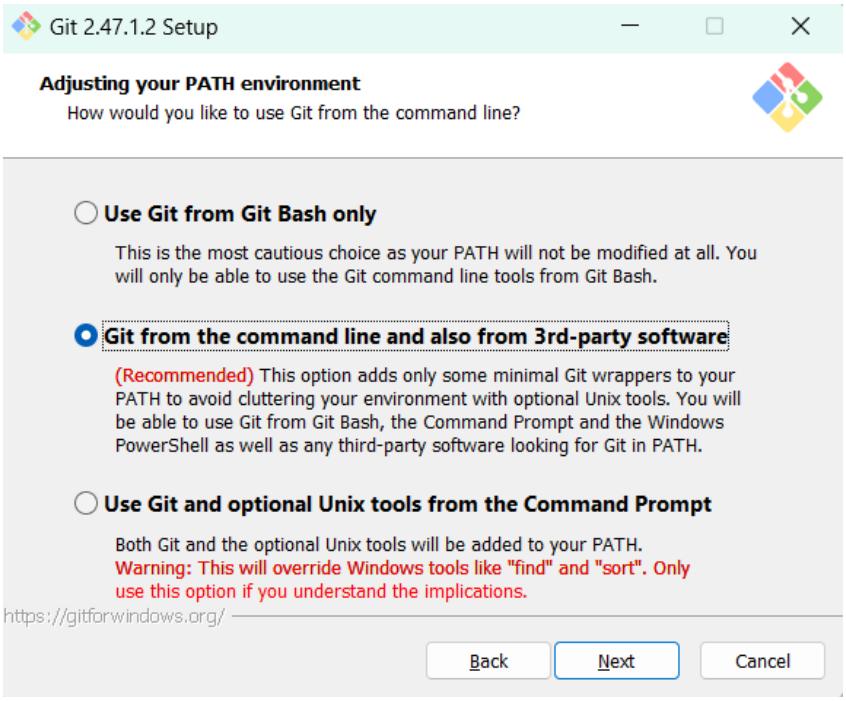
- Vim is a text editor usually found in UNIX and Apple OS X and usually known as "vi".



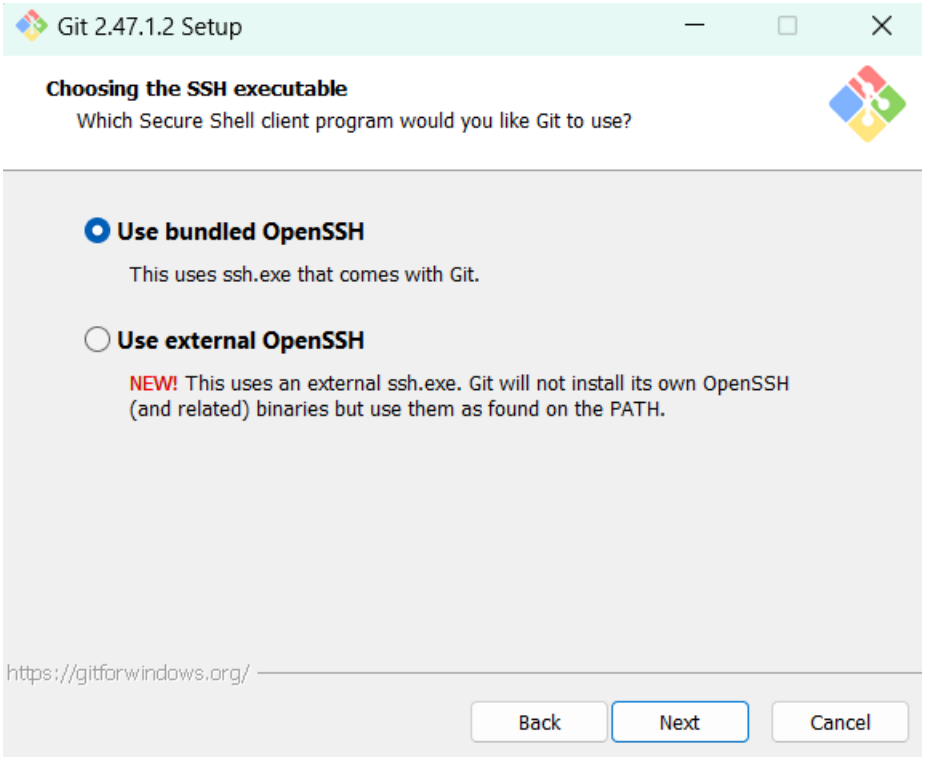
- Select a branch name. you can use the default ‘Master’ or decide to use another name instead.

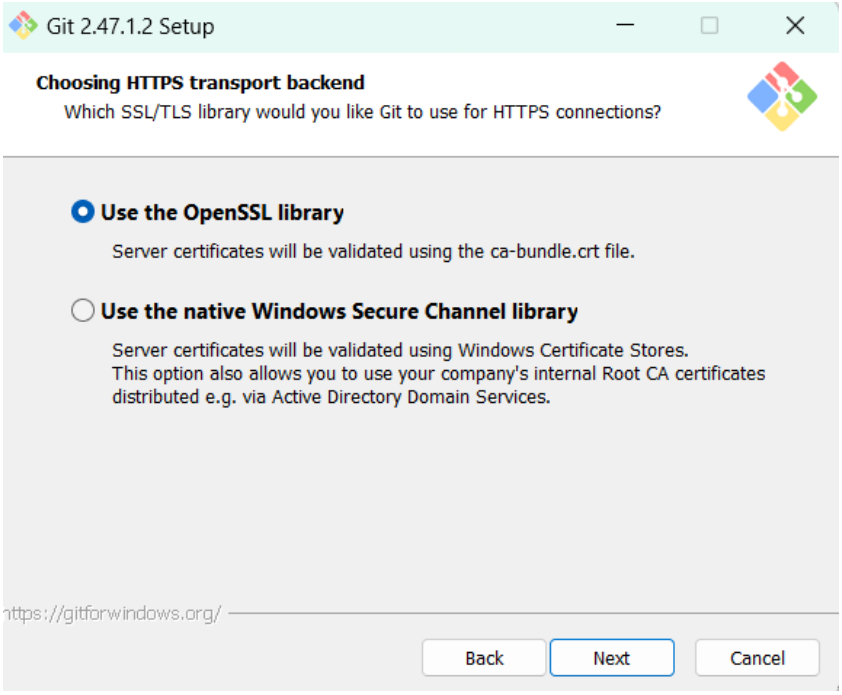


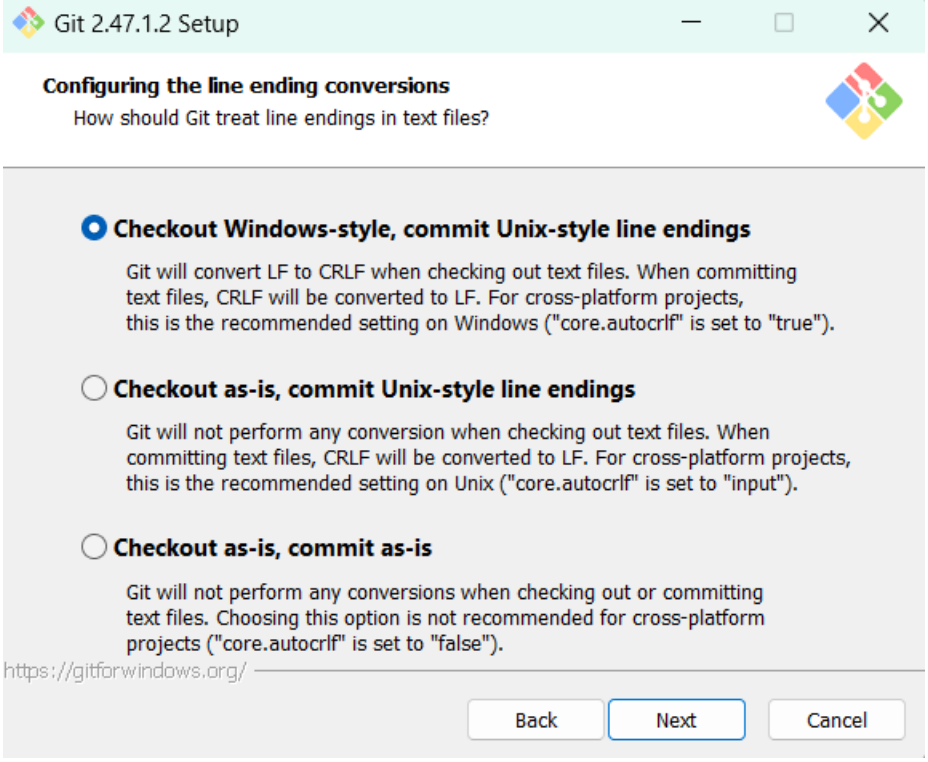
- Adjust and choose your paths enivironment.

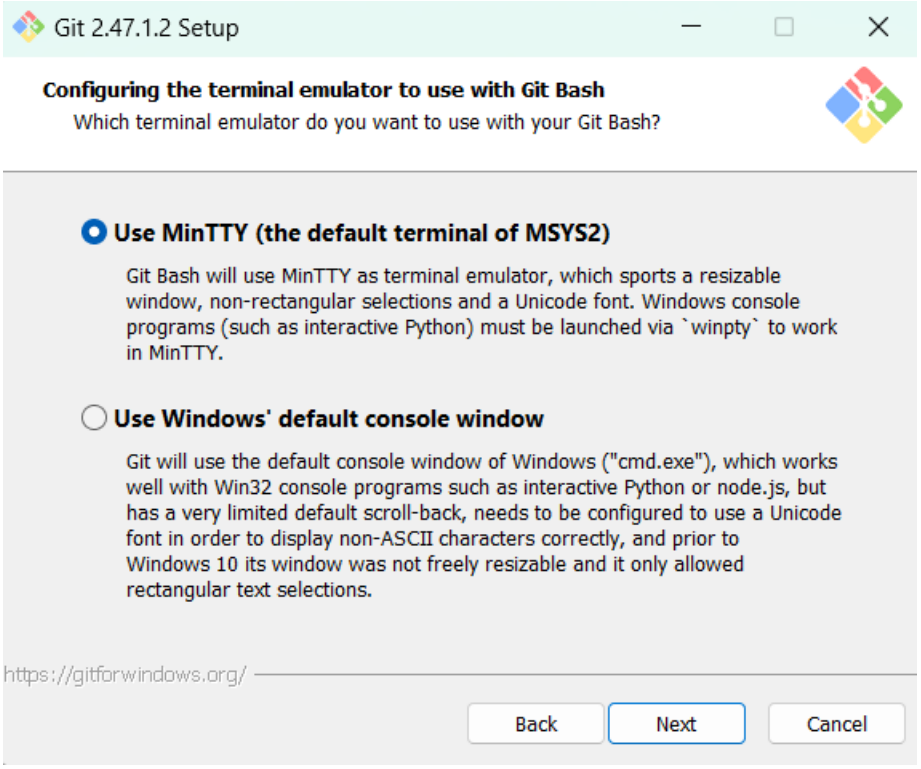


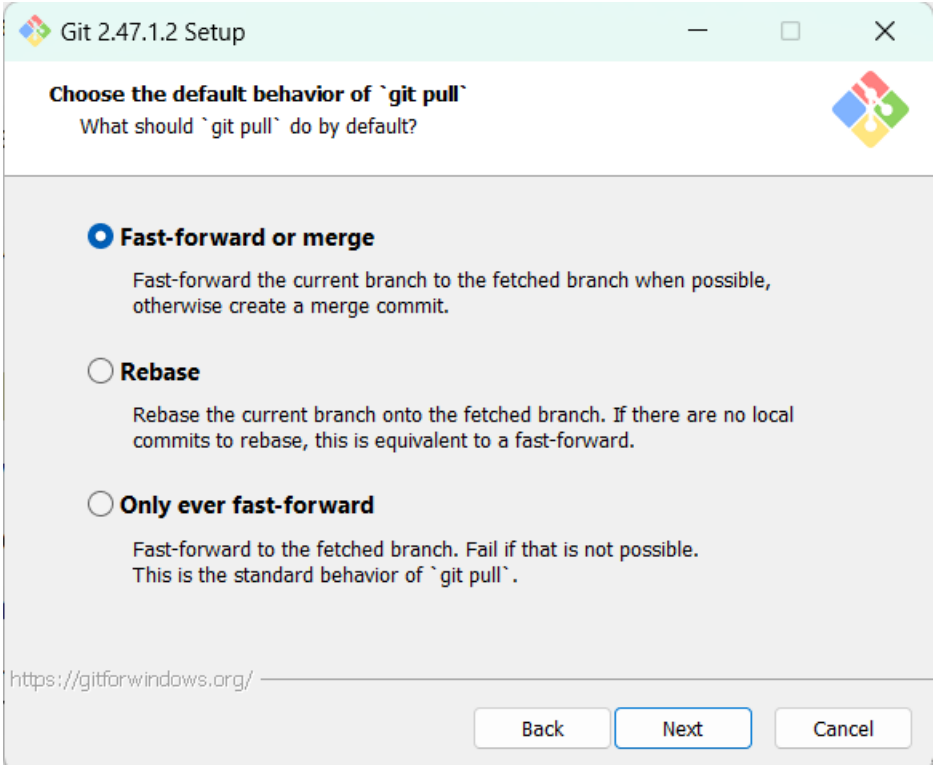
- Choose SSH (Secure Shell Host). Here we are choosing OpenSSH that comes with Git.

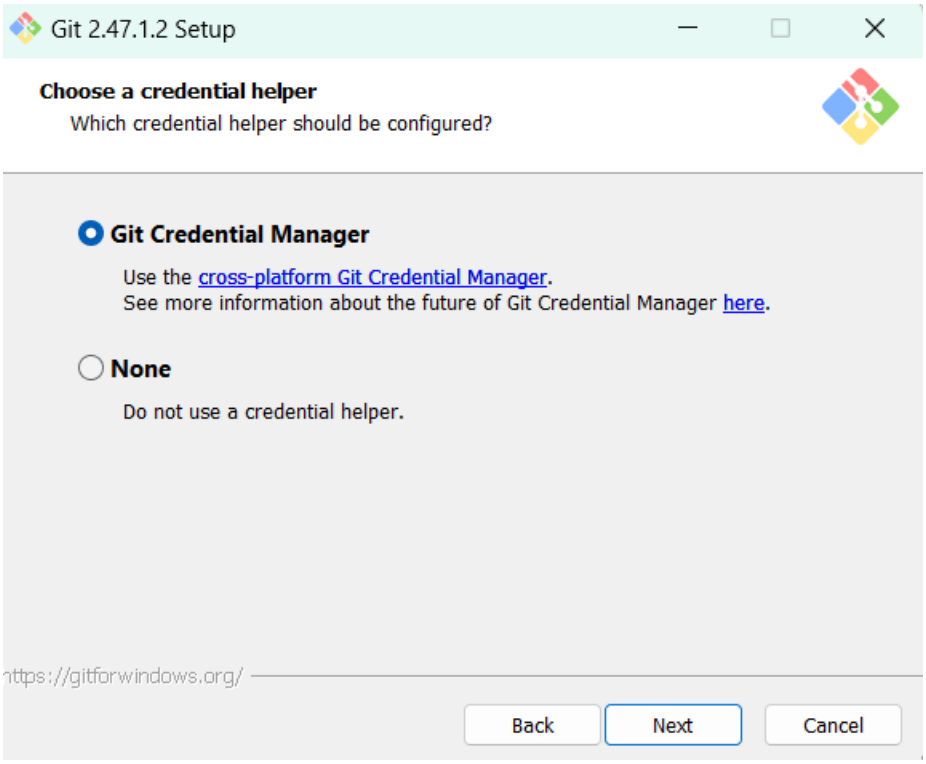


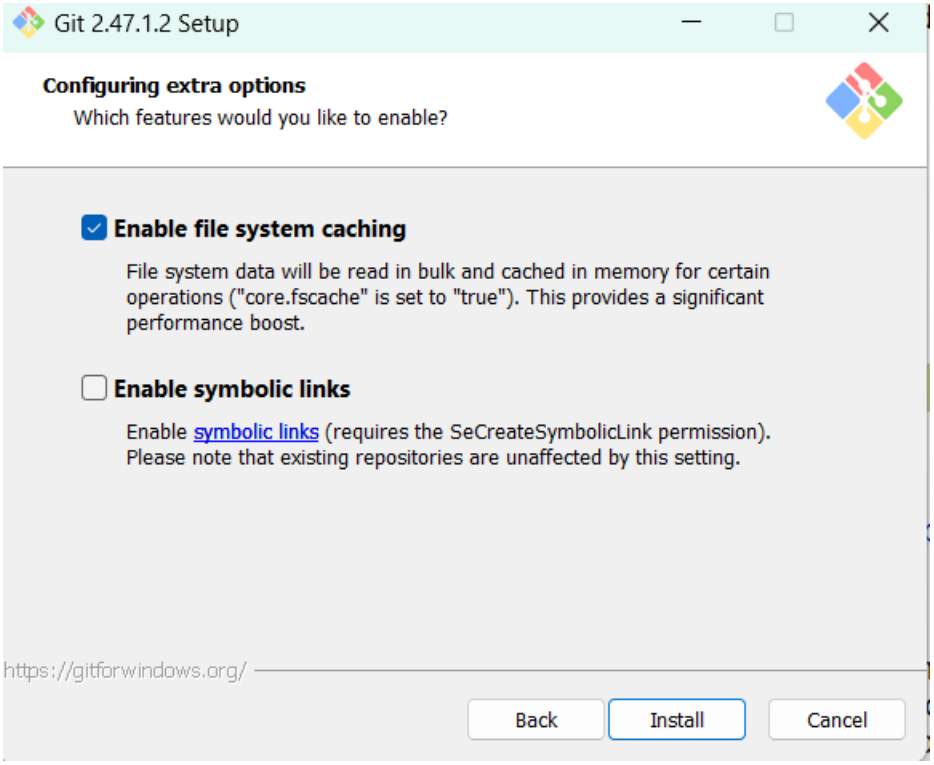


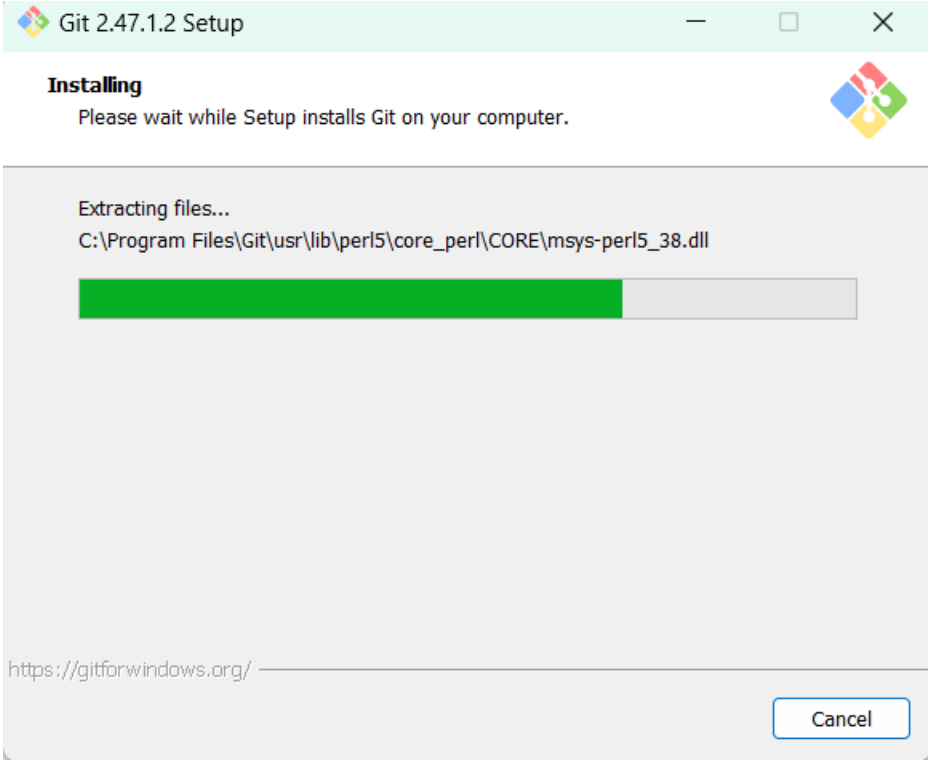

Git 2.47.1.2 Setup
Configuring the line ending conversions
How should Git treat line endings in text files?
Checkout Windows-style, commit Unix-style line endings
Git will convert LF to CRLF when checking out text files. When committing text files, CRLF will be converted to LF. For cross-platform projects, this is the recommended setting on Windows ("core.autocrlf" is set to "true").
Checkout as-is, commit Unix-style line endings
Git will not perform any conversion when checking out text files. When committing text files, CRLF will be converted to LF. For cross-platform projects, this is the recommended setting on Unix ("core.autocrlf" is set to "input").
Checkout as-is, commit as-is
Git will not perform any conversions when checking out or committing text files. Choosing this option is not recommended for cross-platform projects ("core.autocrlf" is set to "false").
https://gitforwindows.org/
Back
Next
Cancel

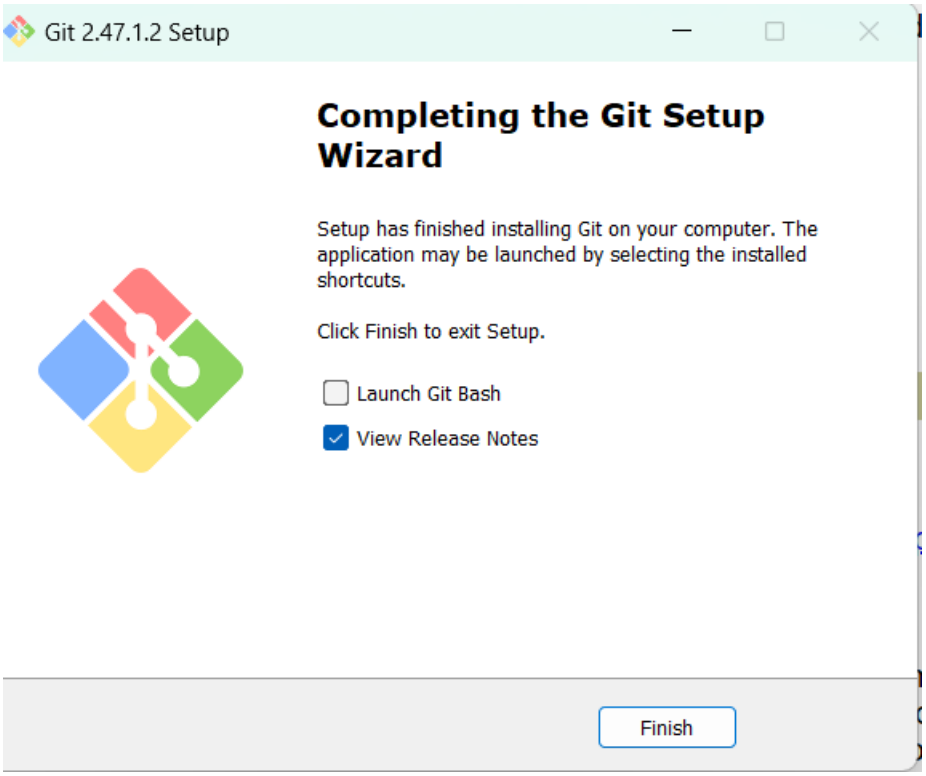

Git 2.47.1.2 Setup
Configuring the terminal emulator to use with Git Bash
Which terminal emulator do you want to use with your Git Bash?
Use MinTTY (the default terminal of MSYS2)
Git Bash will use MinTTY as terminal emulator, which sports a resizable window, non-rectangular selections and a Unicode font. Windows console programs (such as interactive Python) must be launched via `winpty` to work in MinTTY.
Use Windows' default console window
Git will use the default console window of Windows ("cmd.exe"), which works well with Win32 console programs such as interactive Python or node.js, but has a very limited default scroll-back, needs to be configured to use a Unicode font in order to display non-ASCII characters correctly, and prior to Windows 10 its window was not freely resizable and it only allowed rectangular text selections.
https://gitforwindows.org/
Back
Next
Cancel


Git 2.47.1.2 Setup
Choose the default behavior of `git pull`
What should `git pull` do by default?
Fast-forward or merge
Fast-forward the current branch to the fetched branch when possible, otherwise create a merge commit.
Rebase
Rebase the current branch onto the fetched branch. If there are no local commits to rebase, this is equivalent to a fast-forward.
Only ever fast-forward
Fast-forward to the fetched branch. Fail if that is not possible. This is the standard behavior of `git pull`.
https://gitforwindows.org/
Back
Next
Cancel

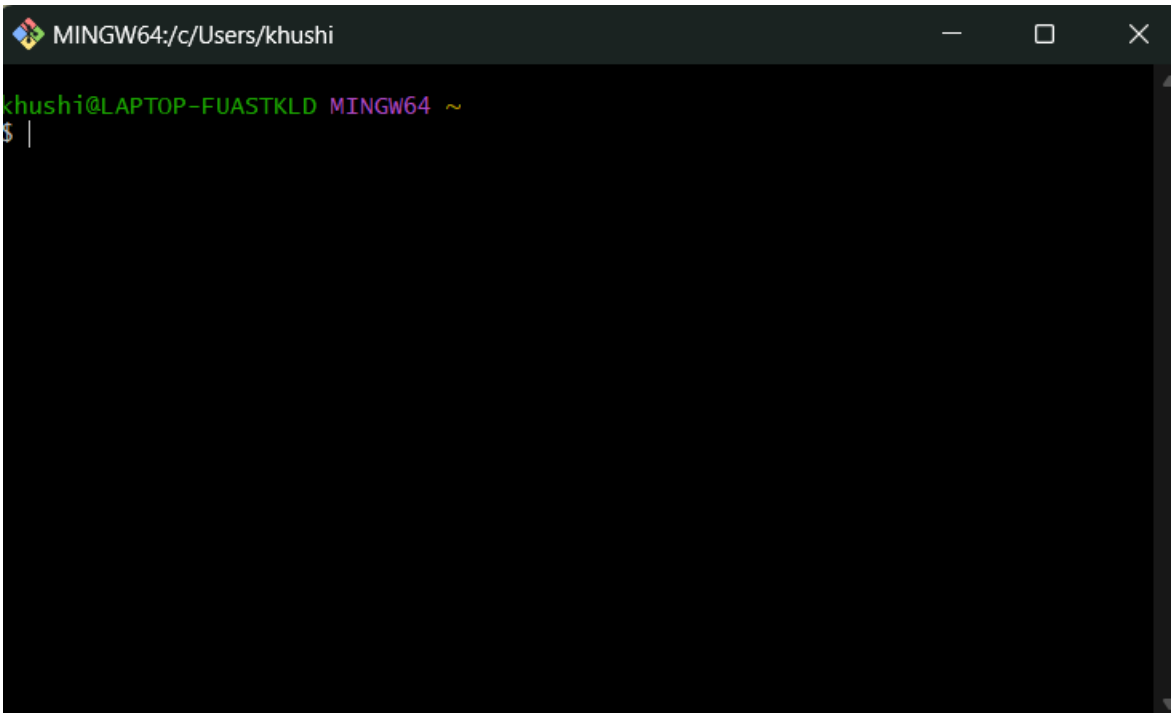

Git 2.47.1.2 Setup
Choose a credential helper
Which credential helper should be configured?
Git Credential Manager
Use the cross-platform Git Credential Manager.
See more information about the future of Git Credential Manager here.
None
Do not use a credential helper.
https://gitforwindows.org/
Back
Next
Cancel


Git 2.47.1.2 Setup
Configuring extra options
Which features would you like to enable?
Enable file system caching
File system data will be read in bulk and cached in memory for certain operations ("core.fscache" is set to "true"). This provides a significant performance boost.
Enable symbolic links
Enable symbolic links (requires the SeCreateSymbolicLink permission). Please note that existing repositories are unaffected by this setting.
https://gitforwindows.org/
Back
Install
Cancel


Git 2.47.1.2 Setup
Installing
Please wait while Setup installs Git on your computer.
Extracting files...
C:\Program Files\Git\usr\lib\perl5\core_perl\CORE\msys-perl5_38.dll
https://gitforwindows.org/
Cancel

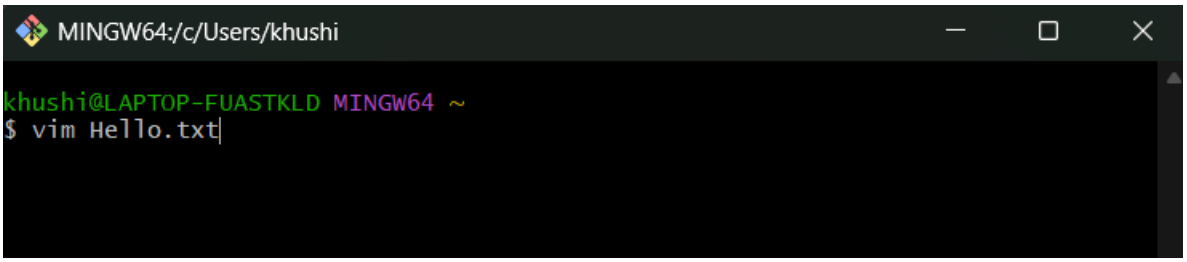


- Open GIT Bash using the Start menu. A terminal will open like the one shown.



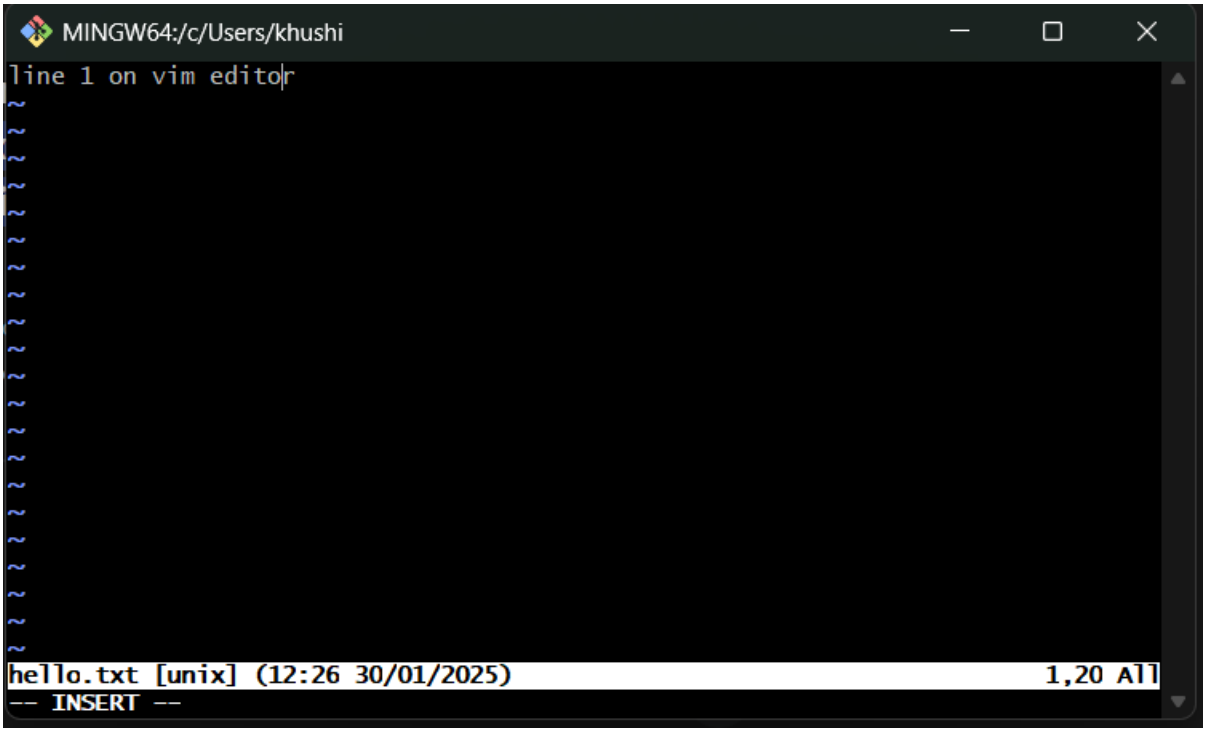
VIM COMMAND

- Vim is the command used to make a file. Here we are making a text file called ‘Hello.txt’. Click enter to create the file.

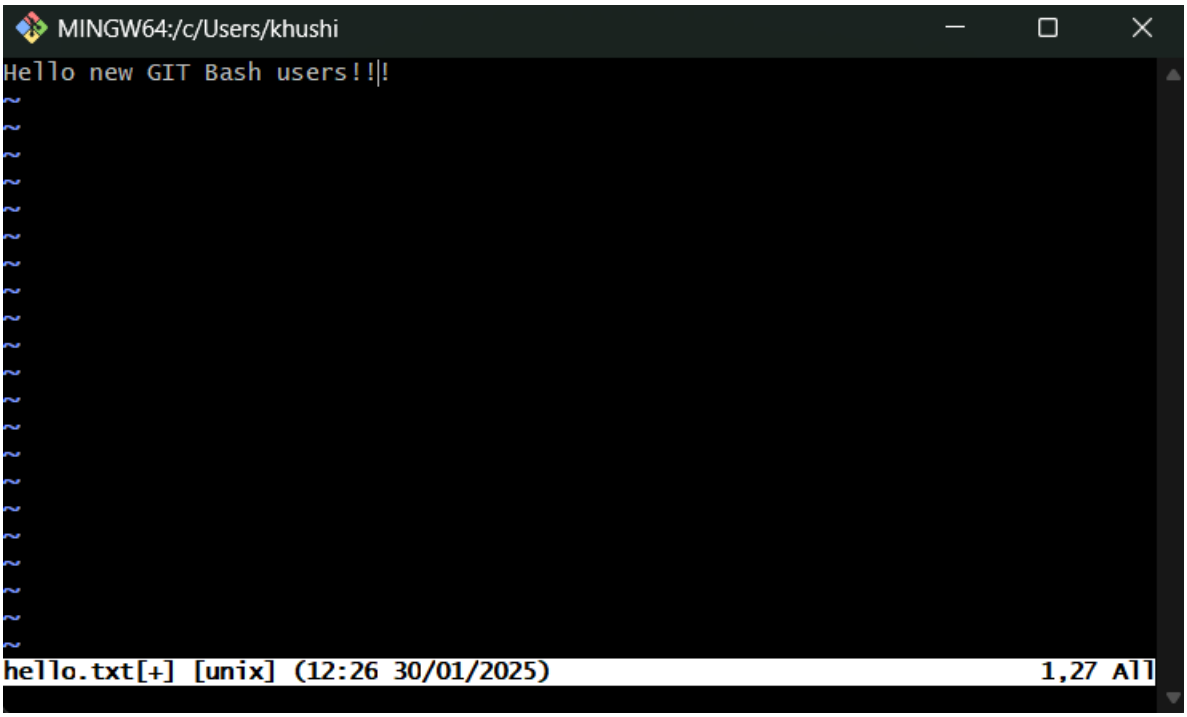


- The following text editor appears. You can’t type any text in the editor unless you enter ‘Insert’ mode. To enter insert mode click on ‘i’ keyboard key.

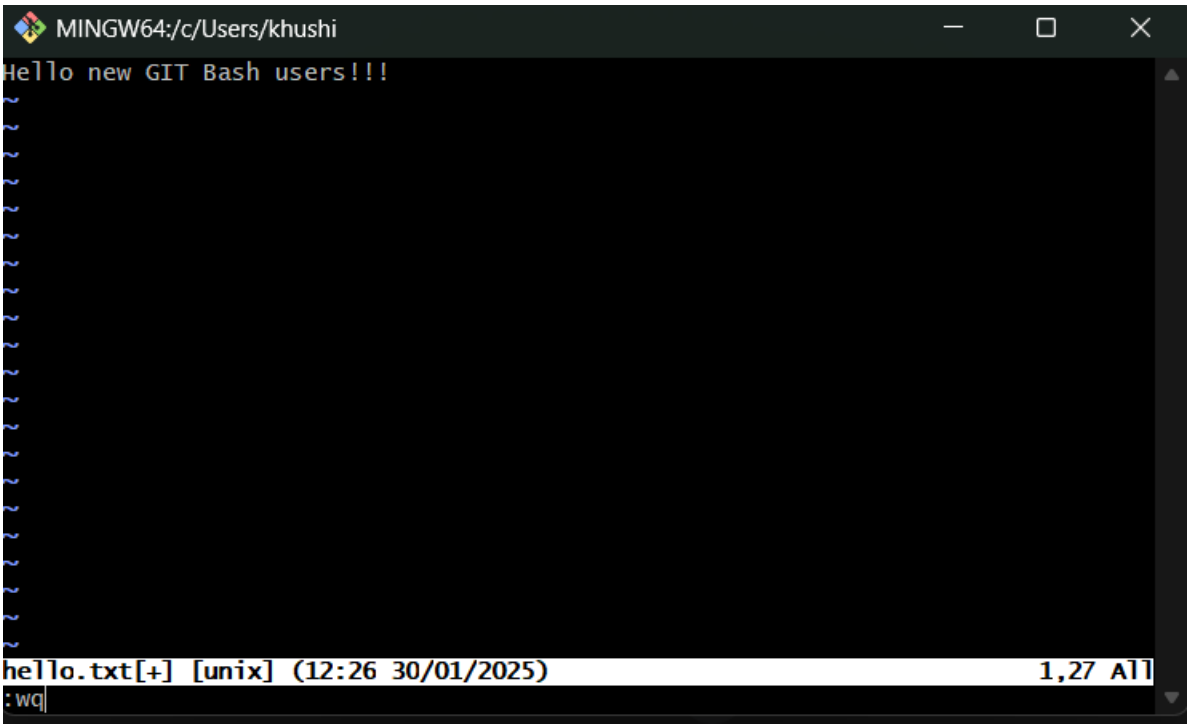
- Now we have entered insert mode, where we can edit the text in the editor. Insert mode is indicated at the bottom of the editor. You can now enter text into your file.



- Once you finished, click the esc key to exit insert mode.

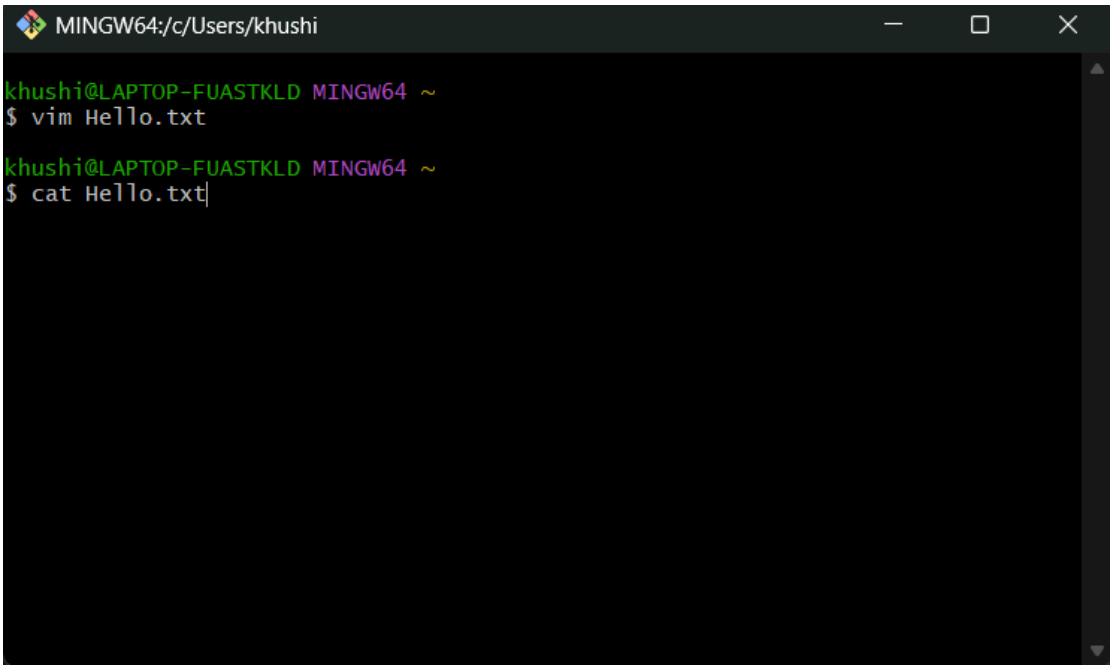


- Type ':wq' to save and exit.

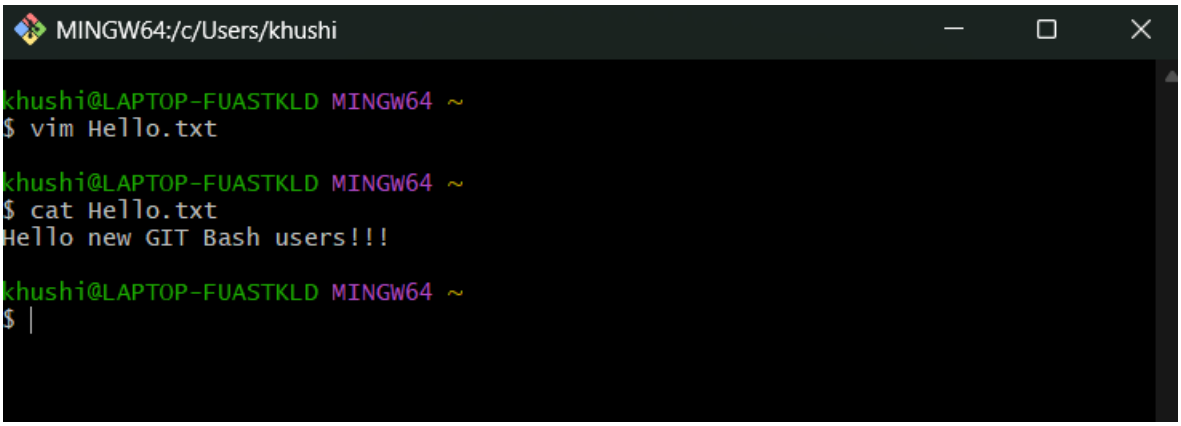


CAT COMMAND

- To display / print the data from a file you have made, we use the 'cat' command.



- After typing in the name of the file, and clicking enter the data within the file is displayed.



ACTIVITY-2

GIT BASH COMMANDS

PWD COMMAND (PRESENT WORKING DIRECTORY)

- The pwd command in GIT Bash allows us to get our current location / path in the system. Right now we are in c drive, in the users folder's file khushi.

LS COMMAND (LIST)

- The ls command lists all the files in the current location.

```
khushi@LAPTOP-FUASTKLD MINGW64 ~
$ ls
 AppData/
'Application Data'@
 Contacts/
 Cookies@
 Desktop/
 Documents/
 Downloads/
 Favorites/
 Jedi/
 Links/
'Local Settings'@
 Music/
'My Documents'@
 NTUSER.DAT
 NTUSER.DAT{d4887892-e3c3-11ef-803b-b5621dbc74cf}.TM.blf
 NTUSER.DAT{d4887892-e3c3-11ef-803b-b5621dbc74cf}.TMContainer00000000000000000000
1.regtrans-ms
 NTUSER.DAT{d4887892-e3c3-11ef-803b-b5621dbc74cf}.TMContainer00000000000000000000
2.regtrans-ms
 NetHood@
 OneDrive/
 Pictures/
 PrintHood@
 Recent@
'Saved Games'/
 Searches/
 SendTo@
'Start Menu'@
 Templates@
 Videos/
'VirtualBox VMs'/
 anaconda3/
 ansel/
 claudesoilmoisture.py
 cow.py
 emojize.py
 figlet.py
 hello.txt
 myenv/
 ntuser.dat.LOG1
 ntuser.dat.LOG2
 ntuser.ini
 usb_driver/
 wifi_monitor.log
 wifi_report.txt

khushi@LAPTOP-FUASTKLD MINGW64 ~
$
```

CD COMMAND

- The cd command allows us to move into / access any file in the current location. The command should be followed by the file / location you want to access.

```
khushi@LAPTOP-FUASTKLD MINGW64 ~
$ cd Desktop

khushi@LAPTOP-FUASTKLD MINGW64 ~/Desktop
$ pwd
/c/Users/khushi/Desktop

khushi@LAPTOP-FUASTKLD MINGW64 ~/Desktop
$
```

MKDIR COMMAND

- This makes a file in your current location. The command is followed by a file name that is to be allotted.

```
khushi@LAPTOP-FUASTKLD MINGW64 ~/Desktop
$ mkdir SourceCodeManagement

khushi@LAPTOP-FUASTKLD MINGW64 ~/Desktop
$ ls
 Anaconda3-2024.06-1-Windows-x86_64.exe*   GAOMONTablet.lnk*                    Netflix.lnk*
 Brave.lnk*                                Games/                              Obsidian.lnk*
'C files'/                                 'Khushi (Person 1) - Chrome.lnk'*   Python/
 C++/                                      'Legion Game Shop.lnk'*            'Python Hand Gesture Recognition.pdf'
'Cry of Fear.url'                          'Microsoft Edge.lnk'*               SourceCodeManagement/
'DELTARUNE (Chapter 1 & 2 DEMO).url'       'MinGW Installer.lnk'*             'Visual Studio Code.lnk'*
```

RMDIR COMMAND

- This command is used to remove any file of your choice from the current location.

```
khushi@LAPTOP-FUASTKLD MINGW64 ~/Desktop
$ rmdir SourceCodeManagement

khushi@LAPTOP-FUASTKLD MINGW64 ~/Desktop
$ ls
 Anaconda3-2024.06-1-Windows-x86_64.exe*  'DELTARUNE (Chapter 1 & 2 DEMO).url'  'Microsoft Edge.lnk'*   'Python Hand Gesture Recognition.pdf'
 Brave.lnk*                                GAOMONTablet.lnk*                   'MinGW Installer.lnk'*  'Visual Studio Code.lnk'*
'C files'/                                 Games/                               Netflix.lnk*            advsimpleclaudesoilmoisture.py
 C++/                                      'Khushi (Person 1) - Chrome.lnk'*    Obsidian.lnk*           ctrial.c
'Cry of Fear.url'                          'Legion Game Shop.lnk'*              Python/                 desktop.ini
```

'CD ..' COMMAND (THERE IS A SPACE IN BETWEEN)

It takes you one step back in your current path location. Here we went from the desktop back to khushi (under users).

```
khushi@LAPTOP-FUASTKLD MINGW64 ~/Desktop
$ cd ..

khushi@LAPTOP-FUASTKLD MINGW64 ~
$ pwd
/c/Users/khushi
```

GIT INIT (REPOSITORY CREATION / MASTER FILE)

This convert the current folder into a master folder / GIT Hub repository format. This cannot be deleted normally.

```
khushi@LAPTOP-FUASTKLD MINGW64 ~/Desktop/hello
$ pwd
/c/Users/khushi/Desktop/hello

khushi@LAPTOP-FUASTKLD MINGW64 ~/Desktop/hello
$ git init
Initialized empty Git repository in C:/Users/khushi/Desktop/hello/.git/

khushi@LAPTOP-FUASTKLD MINGW64 ~/Desktop/hello (master)
$ |
```

LS -AH COMMAND

This displays hidden files from the list of directories under your current location. The ‘ ./ ‘ and ‘../ ‘ are hidden files here under c drive.

```
khushi@LAPTOP-FUASTKLD MINGW64 ~
$ ls -ah
 ./                    Documents/
 ../                   Downloads/
 .VirtualBox/          Favorites/
 .anaconda/            Jedi/
```

Under the master directory or repository file, the following are the hidden files.

```
khushi@LAPTOP-FUASTKLD MINGW6
$ ls -ah
./  ../  .git/
```

COMMITTING IN GIT BASH

- Commits can be thought of as snapshots or milestones along the timeline of a Git project. Commits are created with the git commit command to capture the state of a project at that point in time.
- For a start we can try committing two python files made using the vim editor in git bash (a " .py " file extension was added with the name for converting it into a python file). These were made under the master folder.
- Using the 'ls' command under the master repository folder, we can find the two python files.

```
khushi@LAPTOP-FUASTKLD MINGW6
$ ls
add.py  hello.py
```

- To commit both these files to the repository we use the command line :

  git commit -m "hello.py and add.py"

- If your identity wasn't established with git bash, it will ask for your email id and name for identification.
- To enter the information use:
  git config --global user.name "Khushi Mhamane"
  git config --global user.email khushimhamane@gmail.com

This will allow you to establish your identity.

```
khushi@LAPTOP-FUASTKLD MINGW64 ~/Desktop/4127/pythoncode (master)
$ git commit -m "hello.py and add.py"
Author identity unknown

*** Please tell me who you are.

Run

  git config --global user.email "you@example.com"
  git config --global user.name "Your Name"

to set your account's default identity.
Omit --global to set the identity only in this repository.

fatal: unable to auto-detect email address (got 'khushi@LAPTOP-FUASTKLD.(none)')

khushi@LAPTOP-FUASTKLD MINGW64 ~/Desktop/4127/pythoncode (master)
$ git config --global user.name "Khushi Mhamane"

khushi@LAPTOP-FUASTKLD MINGW64 ~/Desktop/4127/pythoncode (master)
$ git config --global user.email khushimhamane@gmail.com
```

- If you try to commit your files again you will see that you have to "add " your files to the master repository first.
- Use the command line:

  git add hello.py (filenames)

  git add add.py (filenames)

- After adding the files, now you can commit them.

```
khushi@LAPTOP-FUASTKLD MINGW64 ~/Desktop/4127/pythoncode (master)
$ git commit -m "hello.py and add.py"
On branch master

Initial commit

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

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

```
khushi@LAPTOP-FUASTKLD MINGW64 ~/Desktop/4127/pythoncode (master)
$ git add add.py
warning: in the working copy of 'add.py', LF will be replaced by CRLF the next time Git touches it

khushi@LAPTOP-FUASTKLD MINGW64 ~/Desktop/4127/pythoncode (master)
$ git add hello.py
warning: in the working copy of 'hello.py', LF will be replaced by CRLF the next time Git touches it

khushi@LAPTOP-FUASTKLD MINGW64 ~/Desktop/4127/pythoncode (master)
$ git commit -m "hello.py and add.py"
[master (root-commit) fe5aba2] hello.py and add.py
 2 files changed, 8 insertions(+)
 create mode 100644 add.py
 create mode 100644 hello.py
```

- If a file was edited after committing, they will have to be added again and then committed again. Now an original version is stored and the first modified version will be stored. (as screenshots / version data)
- The ‘ git status ‘ command can help you check the condition of the master folder and also if all the files u wanted to commit have been committed or not.

```
khushi@LAPTOP-FUASTKLD MINGW64 ~/Desktop/4127/pythoncode (master)
$ vim hello.py

khushi@LAPTOP-FUASTKLD MINGW64 ~/Desktop/4127/pythoncode (master)
$ vim add.py

khushi@LAPTOP-FUASTKLD MINGW64 ~/Desktop/4127/pythoncode (master)
$ git commit -m "hello.py and add.py"
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:   add.py
        modified:   hello.py

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

khushi@LAPTOP-FUASTKLD MINGW64 ~/Desktop/4127/pythoncode (master)
$ git add add.py
warning: in the working copy of 'add.py', LF will be replaced by CRLF the next time Git touches it

khushi@LAPTOP-FUASTKLD MINGW64 ~/Desktop/4127/pythoncode (master)
$ git add hello.py
warning: in the working copy of 'hello.py', LF will be replaced by CRLF the next time Git touches it

khushi@LAPTOP-FUASTKLD MINGW64 ~/Desktop/4127/pythoncode (master)
$ git commit -m "hello.py and add.py"
[master ee07bb8] hello.py and add.py
 2 files changed, 8 insertions(+), 2 deletions(-)

khushi@LAPTOP-FUASTKLD MINGW64 ~/Desktop/4127/pythoncode (master)
$ git status
On branch master
nothing to commit, working tree clean
```

GIT LOG COMMAND

- Every re commit get assigned a code for that version of the file. The original file will have a code, the first modified version will have a code, the second modified and so on.
- The git log command allows you to get the code of all the versions (commits) made for that file. Since we committed two files at once, from the two codes we will receive (2 commits done, 1 original and 1 modified) the first code is for the originals of both the files and the second code is the modified version of both the files.
- The command

  git log

  or

  git log --oneline

  would give us the codes. 'one line" makes it simpler

```
khushi@LAPTOP-FUASTKLD MINGW64 ~/Desktop/4127/pythoncode (master)
$ git log
commit ee07bb831b4594c357c42ec41f6592ad15acf600 (HEAD -> master)
Author: Khushi Mhamane <khushimhamane@gmail.com>
Date:   Thu Feb 13 11:49:15 2025 +0530

    hello.py and add.py

commit fe5aba2c55b548500c8d6867c2c880586639dd40
Author: Khushi Mhamane <khushimhamane@gmail.com>
Date:   Thu Feb 13 11:46:26 2025 +0530

    hello.py and add.py

khushi@LAPTOP-FUASTKLD MINGW64 ~/Desktop/4127/pythoncode (master)
$ git log --oneline
ee07bb8 (HEAD -> master) hello.py and add.py
fe5aba2 hello.py and add.py
```

GIT DIFF COMMAND

- This command allows us to compare any two commits using their log codes.
- Using the two codes we received using git log we now type in both the codes beside the git diff command line.

  git diff ee07bb8 fe5aba2

- The two files that were committed are shown separately.

```
khushi@LAPTOP-FUASTKLD MINGW64 ~/Desktop/4127/pythoncode (master)
$ git diff ee07bb8 fe5aba2
diff --git a/add.py b/add.py
index fbda1ff..e5215f1 100644
--- a/add.py
+++ b/add.py
@@ -2,11 +2,4 @@ a=input("Enter a number:")
 b=input("Enter a second number:")
 sum=a+b
 print(f"The sum is: {sum}")
-print("edit")
-
-
-
-
-
-

diff --git a/hello.py b/hello.py
index a327218..ce26f29 100644
--- a/hello.py
+++ b/hello.py
@@ -1,2 +1,3 @@
 name=input("Enter your name:")
-print(f"Hello {name}.")
+id=input("Enter your id:")
+print(f"Hello {name}. This is your id: {id}")
```

ACTIVITY-3

GIT REMOTE

- Checks for all the available linked Github repositories connected to your master repository

```
khushi@LAPTOP-FUASTKLD MINGW64 ~/Desktop/4127/ccodes (master)
$ git remote
```

GIT REMOTE ADD

- Allows you to create a variable that links a Github repository to your repo. Using this variable name (eg: "L-15"), you can push your files to your Github repository.

```
khushi@LAPTOP-FUASTKLD MINGW64 ~/Desktop/4127/ccodes (master)
$ git remote add L-15 "https://github.com/Ishwaltz/L-15K"
```

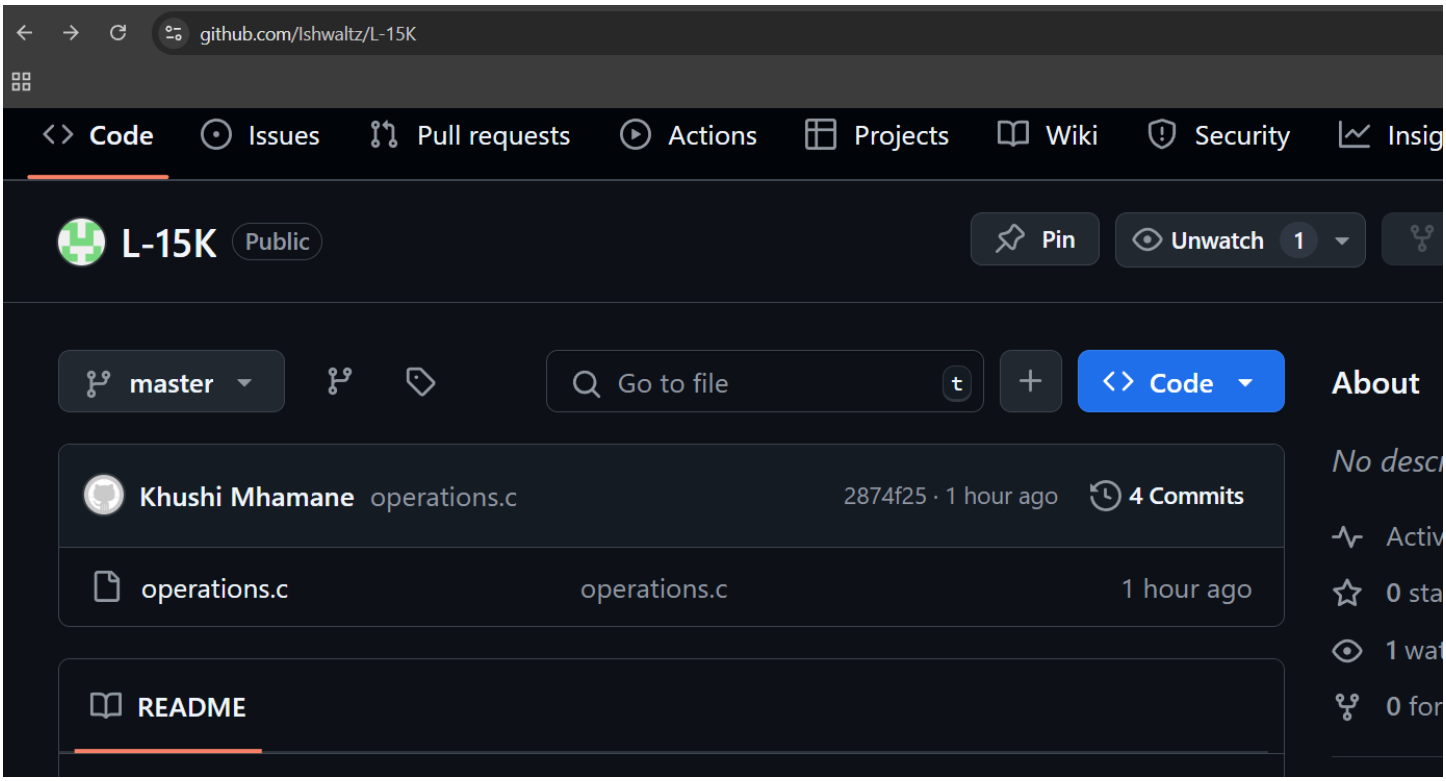
```
khushi@LAPTOP-FUASTKLD MINGW64 ~/Desktop/4127/ccodes (master)
$ git remote
L-15
L-15K
```

GIT PUSH

- This allows you to transfer/push your repository files to your online Github account. You need to specify the variable that contains the link and the branch you wish to push.
- These files will then appear on your profile after pushing.

- You can also view all the commits and changes you have made to the file prior to pushing.

```
khushi@LAPTOP-FUASTKLD MINGW64 ~/Desktop/4127/ccodes (master)
$ git push -u L-15 master
info: please complete authentication in your browser...
Enumerating objects: 12, done.
Counting objects: 100% (12/12), done.
Delta compression using up to 16 threads
Compressing objects: 100% (8/8), done.
Writing objects: 100% (12/12), 1.21 KiB | 248.00 KiB/s, done.
Total 12 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/Ishwaltz/L-15K
 * [new branch]      master -> master
branch 'master' set up to track 'L-15/master'.
```



operations.c

master

1 parent 5caba50 commit 2874f25

Filter files...

operations.c

1 file changed +4 -4 lines changed

Search within code

operations.c +4 -4

```diff
@@ -1,9 +1,9 @@
 #include <stdio.h>
 int main()
 {
-        int a,b,mul;
+        int a,b,div;
         a=3;
-        b=2;
-        mul= a+b;
-        printf("The diff is: %d",&mul);
+        b=3;
+        div= a+b;
+        printf("The div is: %d",&div);
 }
```

operations.c +4 -4

```diff
 #include <stdio.h>
 int main()
 {
-        int a,b,mul;
+        int a,b,div;
         a=3;
-        b=2;
-        mul= a+b;
-        printf("The diff is: %d",&mul);
+        b=3;
+        div= a+b;
+        printf("The div is: %d",&div);
 }
```

ACTIVITY- 4

GIT BRANCH

- Git branch command can let you view the branches available in your repository.
- To make a branch, you write a new branch name following the git branch command.
- Your current branch will be highlighted and have an asterisk '*' displayed beside it.

MAKING A BRANCH

```
khushi@LAPTOP-FUASTKLD MINGW64 ~/Desktop/4127/ccodes (master)
$ git branch bran1
```

VIEWING ALL AVAILABLE BRANCHES

```
khushi@LAPTOP-FUASTKLD MINGW64 ~/Desktop/4127/ccodes (master)
$ git branch
  bran1
* master
```

GIT CHECKOUT

- Git checkout command allows you to change your current branch.

```
khushi@LAPTOP-FUASTKLD MINGW64 ~/Desktop/4127/ccodes (master)
$ git checkout bran1
M       operations.c
Switched to branch 'bran1'

khushi@LAPTOP-FUASTKLD MINGW64 ~/Desktop/4127/ccodes (bran1)
$ ls
a.exe*  operations.c
```

- When you make an additional commit or make changes to a file inside a branch (after branching from main) these changes aren't displayed in the main branch.
- Changes made in the main branch after a branch has been derived from it, wont display itself in the sub-branch.

```
khushi@LAPTOP-FUASTKLD MINGW64 ~/Desktop/4127/ccodes (bran1)
$ vim operations.c

khushi@LAPTOP-FUASTKLD MINGW64 ~/Desktop/4127/ccodes (bran1)
$ git add operations.c
warning: in the working copy of 'operations.c', LF will be r

khushi@LAPTOP-FUASTKLD MINGW64 ~/Desktop/4127/ccodes (bran1)
$ git commit -m"operations.c"
[bran1 463478c] operations.c
```

```
khushi@LAPTOP-FUASTKLD MINGW64 ~/Desktop/4127/ccodes (bran1)
$ git log --oneline
463478c (HEAD -> bran1) operations.c
2874f25 (L-15/master, master) operations.c
5caba50 operations.c
46d7687 operations.c
3fc55d4 operations.c
```

```
khushi@LAPTOP-FUASTKLD MINGW64 ~/Desktop/4127/ccodes (bran1)
$ cat operations.c
#include <stdio.h>
int main()
{
        int a,b,div;
        a=9;
        b=3;
        div= a/b;
        printf("The div is: %d",&div);
}
```

```
khushi@LAPTOP-FUASTKLD MINGW64 ~/Desktop/4127/ccodes (bran1)
$ git checkout master
Switched to branch 'master'
Your branch is up to date with 'L-15/master'.
```

```
khushi@LAPTOP-FUASTKLD MINGW64 ~/Desktop/4127/ccodes (master)
$ cat operations.c
#include <stdio.h>
int main()
{
        int a,b,div;
        a=3;
        b=3;
        div= a+b;
        printf("The div is: %d",&div);
}
```

ACTIVITY- 5

GIT MERGE

- Git merge command allows you to merge a branch into your current branch. This can be done by stating the branch you wish to merge with after the git merge command.

```
khushi@LAPTOP-FUASTKLD MINGW64 ~/Desktop/4127/scmfinalproject/Buildwithai (b1)
$ git checkout master
A       gitignore
Switched to branch 'master'

khushi@LAPTOP-FUASTKLD MINGW64 ~/Desktop/4127/scmfinalproject/Buildwithai (master)
$ git merge b1
```

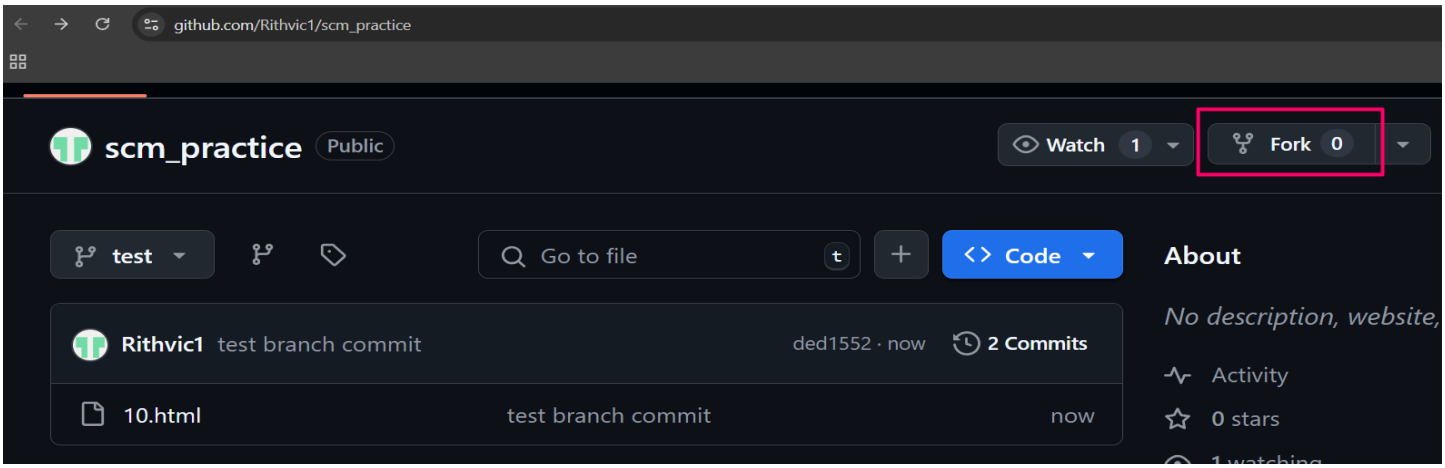
```
khushi@LAPTOP-FUASTKLD MINGW64 ~/Desktop/4127/scmfinalproject/Buildwithai (master)
$ git merge b1
error: Your local changes to the following files would be overwritten by merge:
  gitignore
Merge with strategy ort failed.

khushi@LAPTOP-FUASTKLD MINGW64 ~/Desktop/4127/scmfinalproject/Buildwithai (master)
$ git mergetool --tool=vimdiff
```

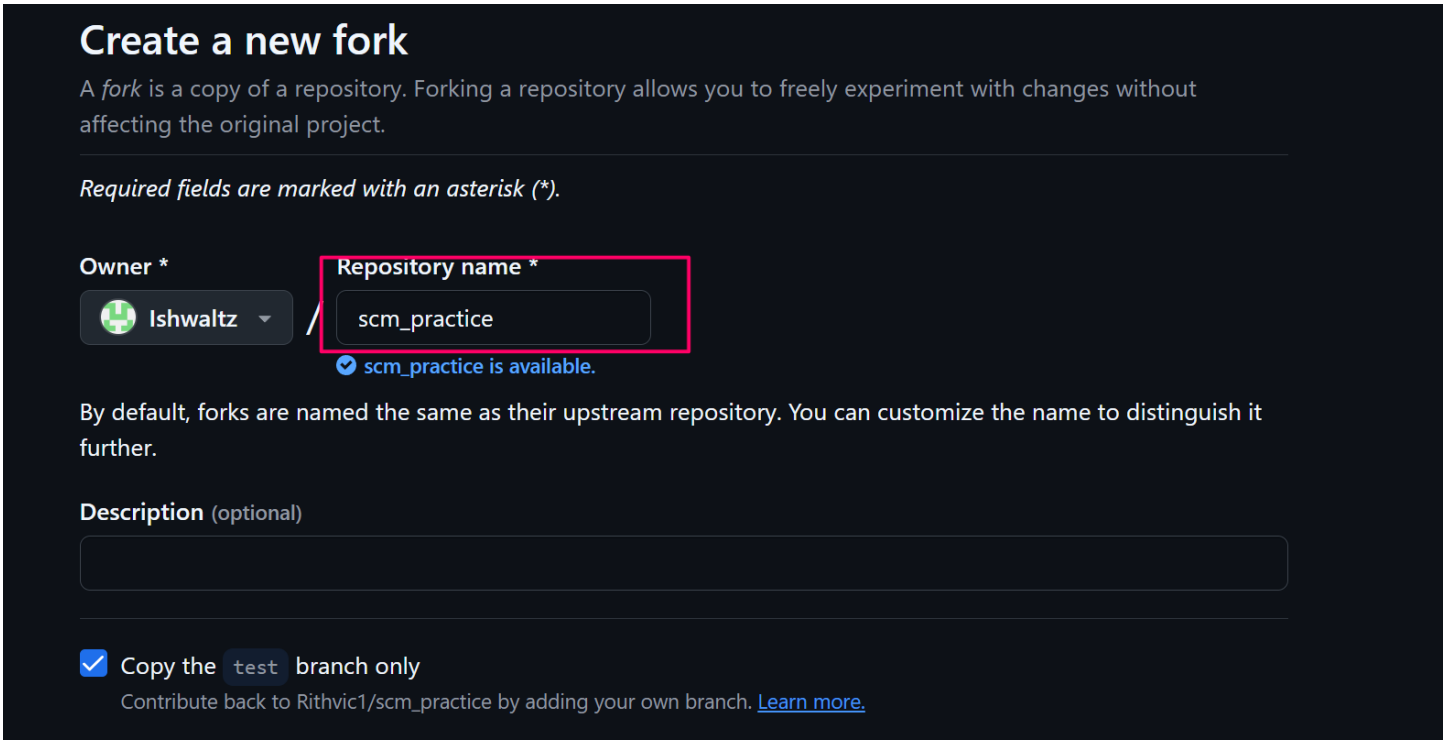
```
Merge made by the 'ort' strategy.
 ReadME.md | 1 +
 gitignore | 1 +
 2 files changed, 2 insertions(+)
 create mode 100644 ReadME.md
 create mode 100644 gitignore
```

ACTIVITY- 6

FORKING A REPOSITORY

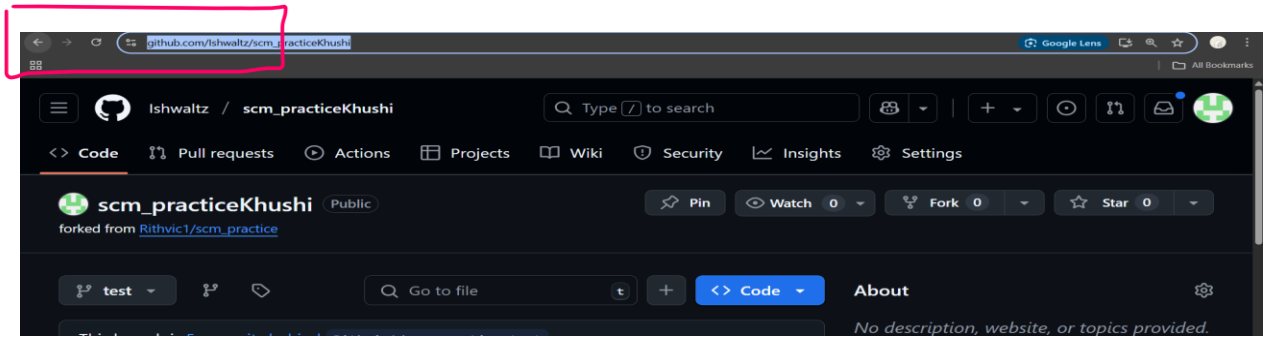
- Forking a repo can be done by clicking on the fork button in the Github repository. This will allow you to have your own repository containing the files of the repository you forked.
- You can name your forked repository with a new name. Users use this to start contributing to other people's repositories.





GIT CLONE

- This allows you to clone a forked repository or any other repository onto your device using the repository's link.
- After cloning the files you can add branches and make changes.
- These changes can be pushed to your forked repository on Github.



```
khushi@LAPTOP-FUASTKLD MINGW64 ~/Desktop/4127/report
$ git clone https://github.com/Ishwaltz/scm_practiceKhushi
Cloning into 'scm_practiceKhushi'...
remote: Enumerating objects: 9, done.
remote: Counting objects: 100% (9/9), done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 9 (delta 0), reused 9 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (9/9), done.

khushi@LAPTOP-FUASTKLD MINGW64 ~/Desktop/4127/report
$ ls
scm_practiceKhushi/
```

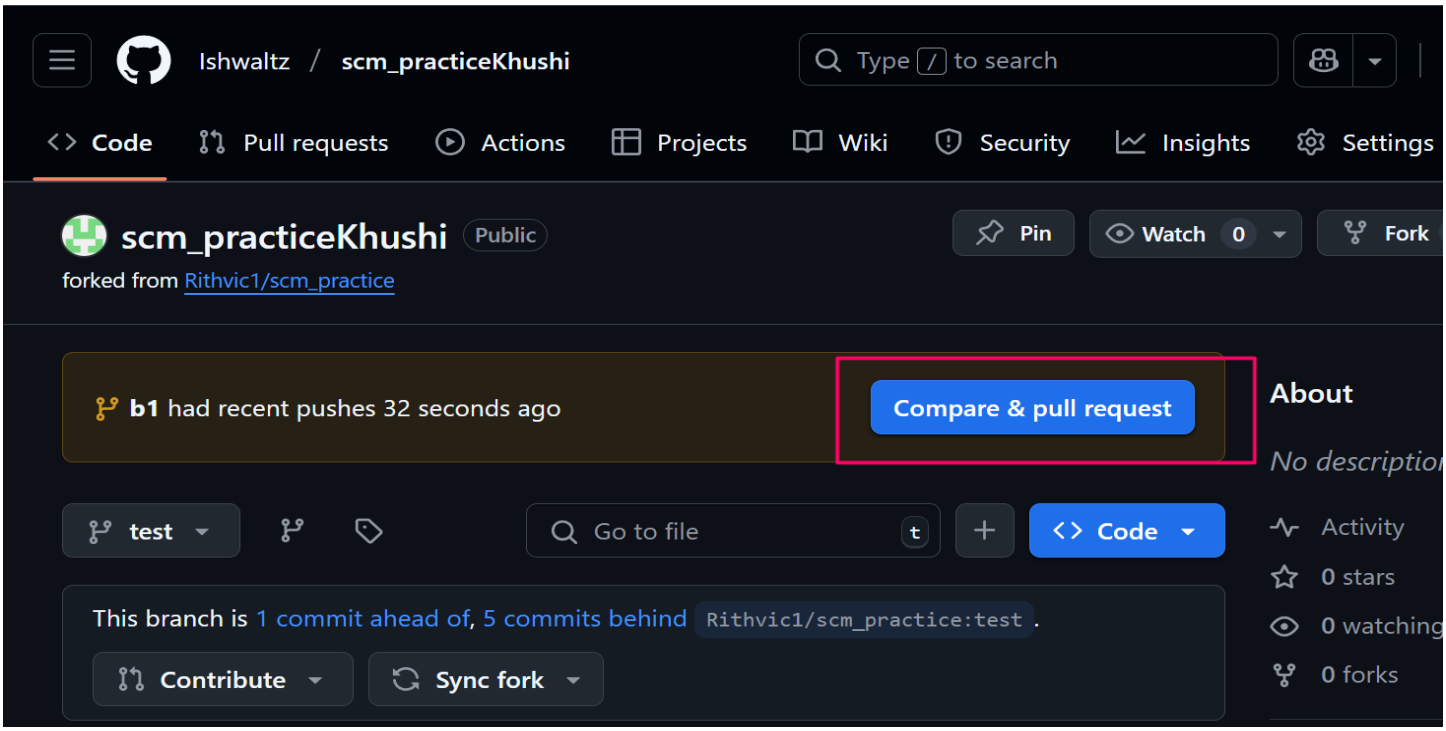
```
khushi@LAPTOP-FUASTKLD MINGW64 ~/Desktop/4127/report
$ cd scm_practiceKhushi/

khushi@LAPTOP-FUASTKLD MINGW64 ~/Desktop/4127/report/scm_practiceKhushi (te
st)
$ ls
10.html
```

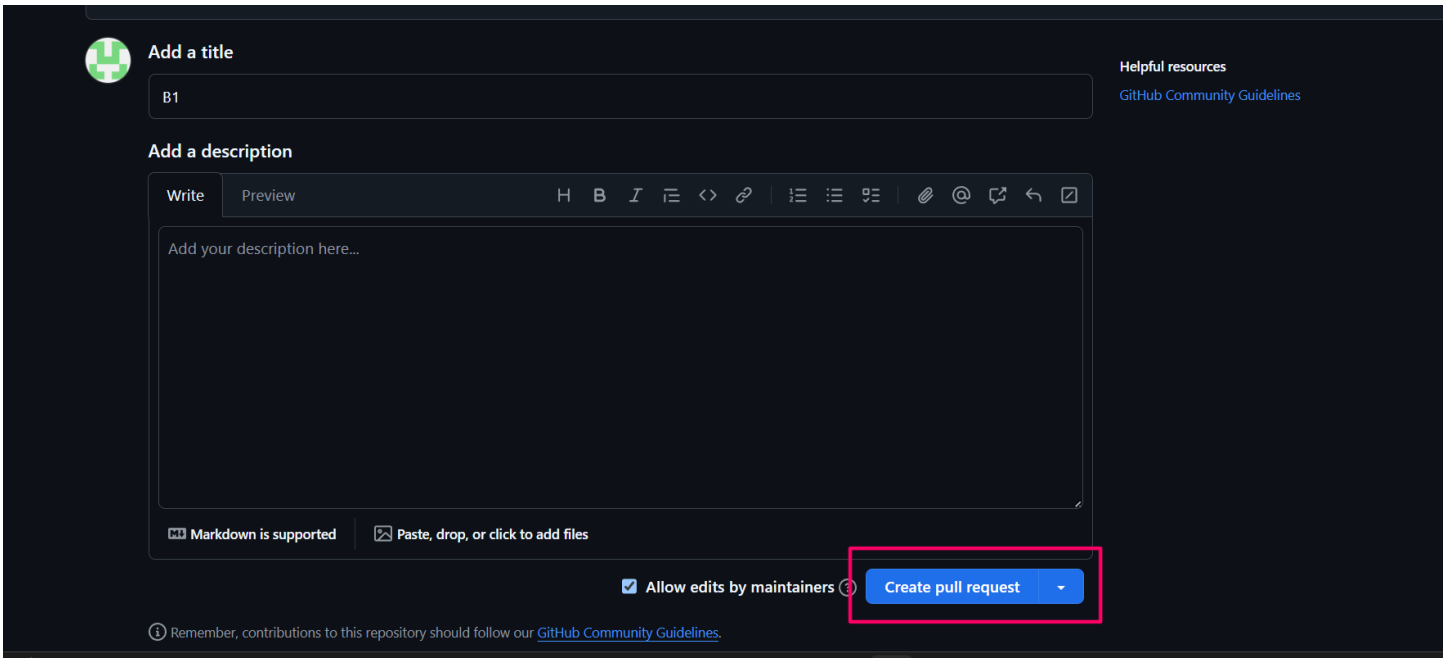
```
khushi@LAPTOP-FUASTKLD MINGW64 ~/Desktop/4127/report/scm_practiceKhushi (test)
$ git branch b1

khushi@LAPTOP-FUASTKLD MINGW64 ~/Desktop/4127/report/scm_practiceKhushi (test)
$ git checkout b1
Switched to branch 'b1'

khushi@LAPTOP-FUASTKLD MINGW64 ~/Desktop/4127/report/scm_practiceKhushi (b1)
$ vim 10.html

khushi@LAPTOP-FUASTKLD MINGW64 ~/Desktop/4127/report/scm_practiceKhushi (b1)
$ git add .

khushi@LAPTOP-FUASTKLD MINGW64 ~/Desktop/4127/report/scm_practiceKhushi (b1)
$ git commit -m "Another commit"
[b1 f96565f] Another commit
 1 file changed, 1 insertion(+)
```

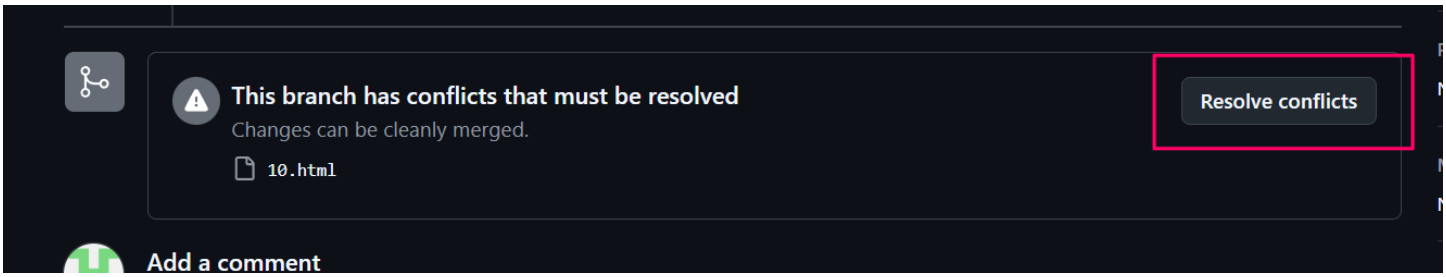
```
khushi@LAPTOP-FUASTKLD MINGW64 ~/Desktop/4127/report/scm_practiceKhushi (b1)
$ git push -u origin b1
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 16 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 294 bytes | 294.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
remote:
remote: Create a pull request for 'b1' on GitHub by visiting:
remote:      https://github.com/Ishwaltz/scm_practiceKhushi/pull/new/b1
remote:
To https://github.com/Ishwaltz/scm_practiceKhushi
 * [new branch]      b1 -> b1
branch 'b1' set up to track 'origin/b1'.
```

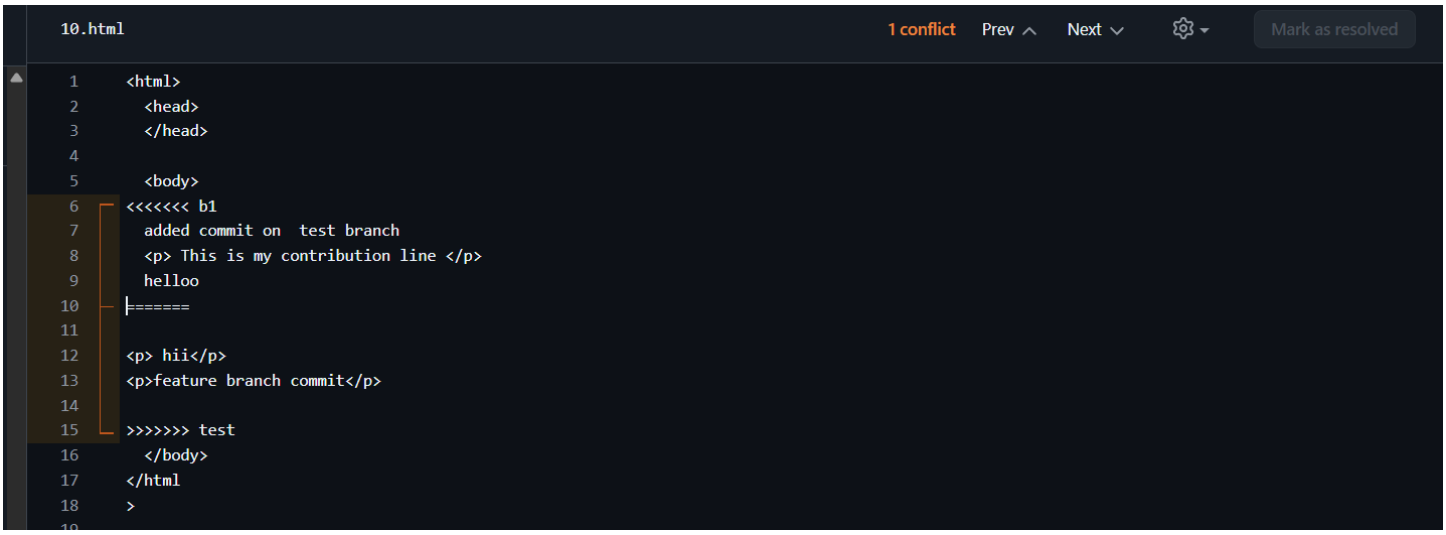
PULL REQUESTS (SEND AND ACCEPT)

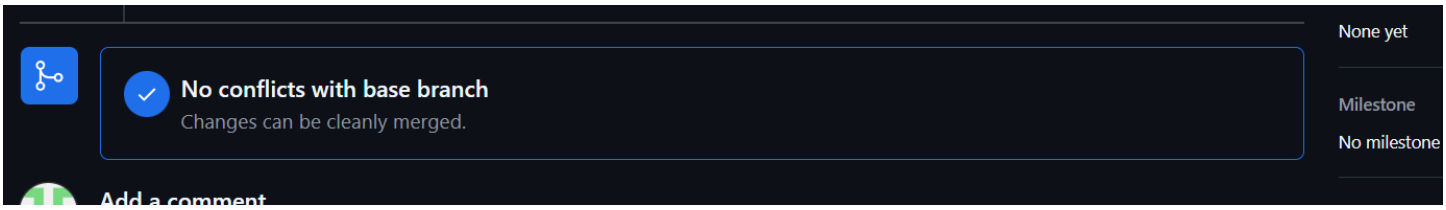
- After you clone a repository and made changes to it, you can then send the owner of the repository that you cloned, a pull request.
- This allows them to view the changes that u made to their code and decide whether to accept and merge your code to their project or not.
- This is how you contribute to others projects through Github.
- While sending a pull request youy may have merge conflicts where the code cant seamlessly integrate. These can be resolved in Github itself or through Gitbash prior.




Add a title
B1
Add a description
Write
Preview
Add your description here...
Markdown is supported
Paste, drop, or click to add files
Allow edits by maintainers
Create pull request
Remember, contributions to this repository should follow our GitHub Community Guidelines.
Helpful resources
GitHub Community Guidelines


This branch has conflicts that must be resolved
Changes can be cleanly merged.
10.html
Resolve conflicts
Add a comment


10.html
1 conflict
Prev
Next
Mark as resolved
<html>
<head>
</head>
<body>
<<<<<<< b1
added commit on test branch
 This is my contribution line
helloo
=======
 hii
feature branch commit
>>>>>>> test
</body>
</html
>



GIT PULL

- This is used to pull all the changes from git hub onto your local computer.