SUBJECT:

SOURCE CODE MANAGEMENT (CS181)

SUBMITTED TO: Dr. MONIT KAPOOR

SUBMITTED BY: DEVANSH DOGRA

ROLL NO.: 2110990422

CLUSTER: BETA

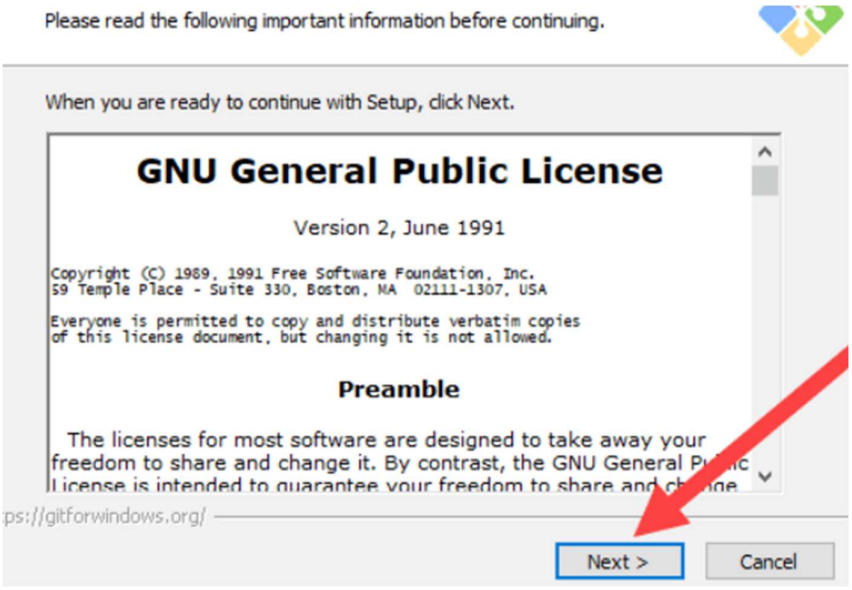
CONTENTS:

TASK 1.1:

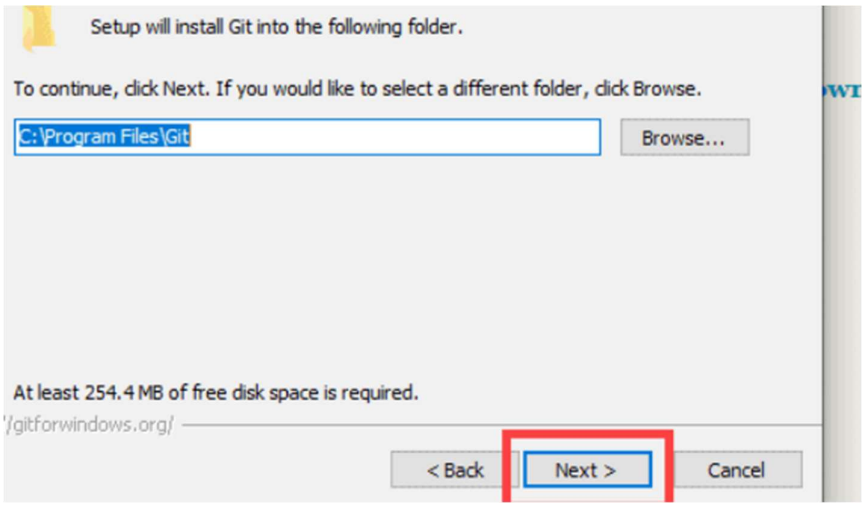
1) SETTING UP GIT CLIENT:

1. Go to the official Git website: https://git-scm.com/downloads
2. Click the download link for Windows and allow the download to complete.

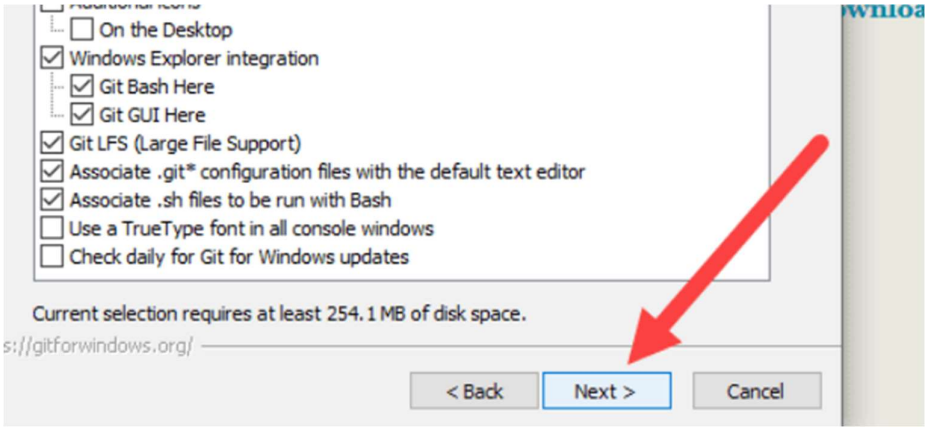
3. Browse to the download location (or use the download shortcut in your browser). Double-click the file to extract and launch the installer.
4. Review the GNU General Public License, and when you're ready to install, click **Next**.



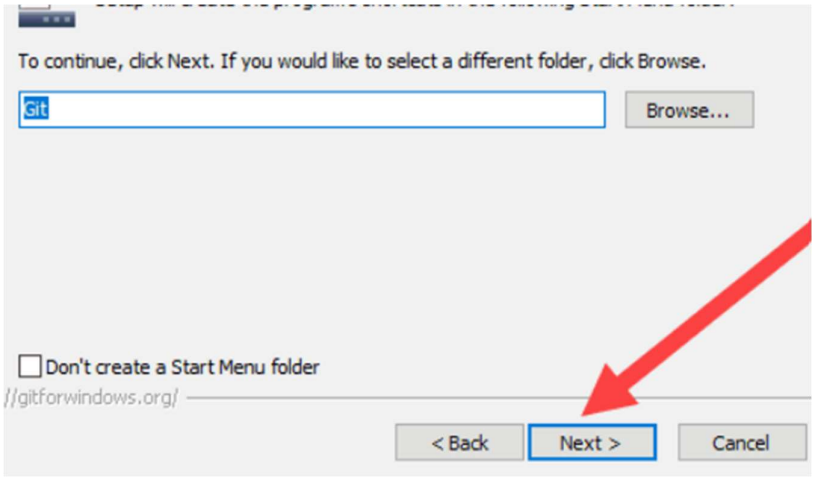
5. The installer will ask you for an installation location. Leave the default, unless you have reason to change it, and click **Next**.



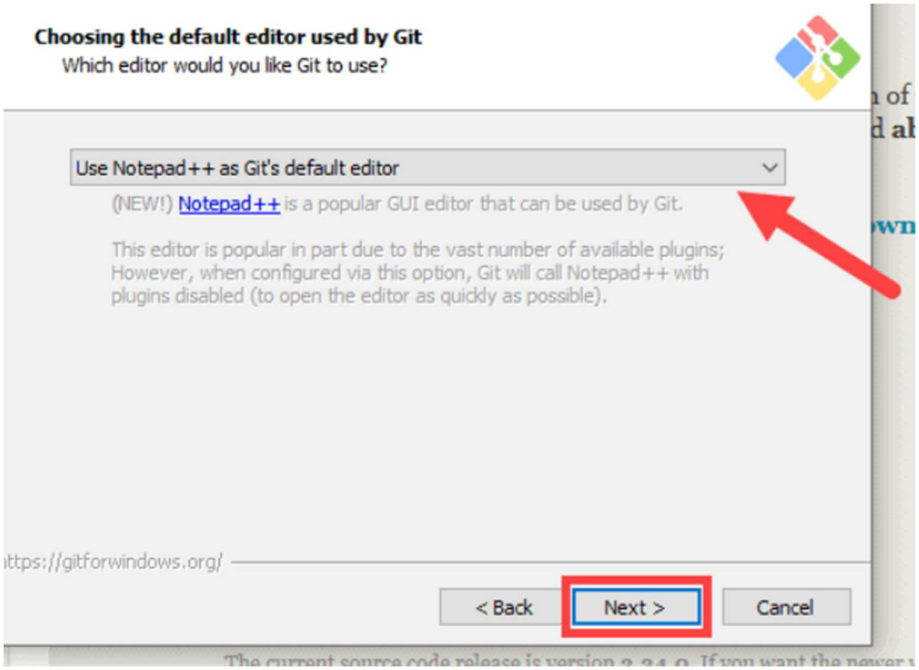
6. A component selection screen will appear. Leave the defaults unless you have a specific need to change them and click **Next**.



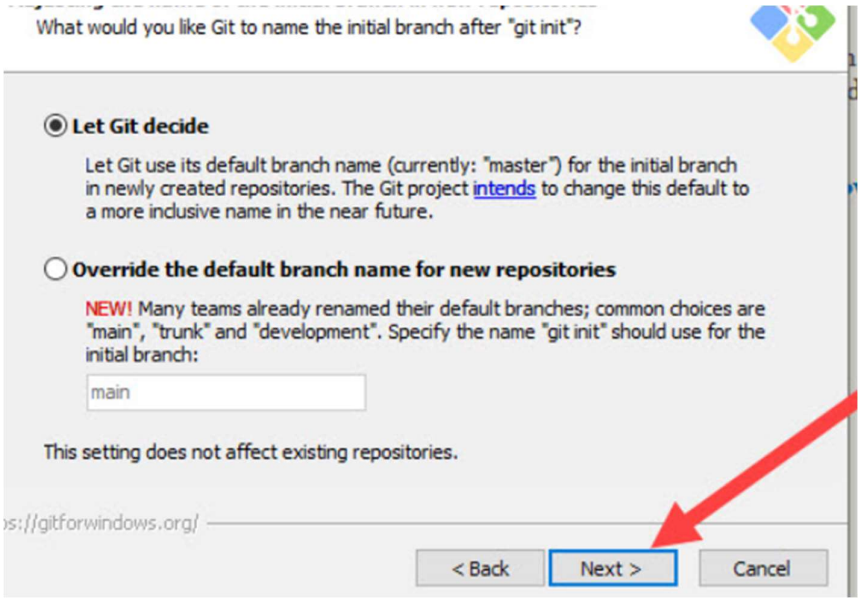
7. The installer will offer to create a start menu folder. Simply click **Next**.



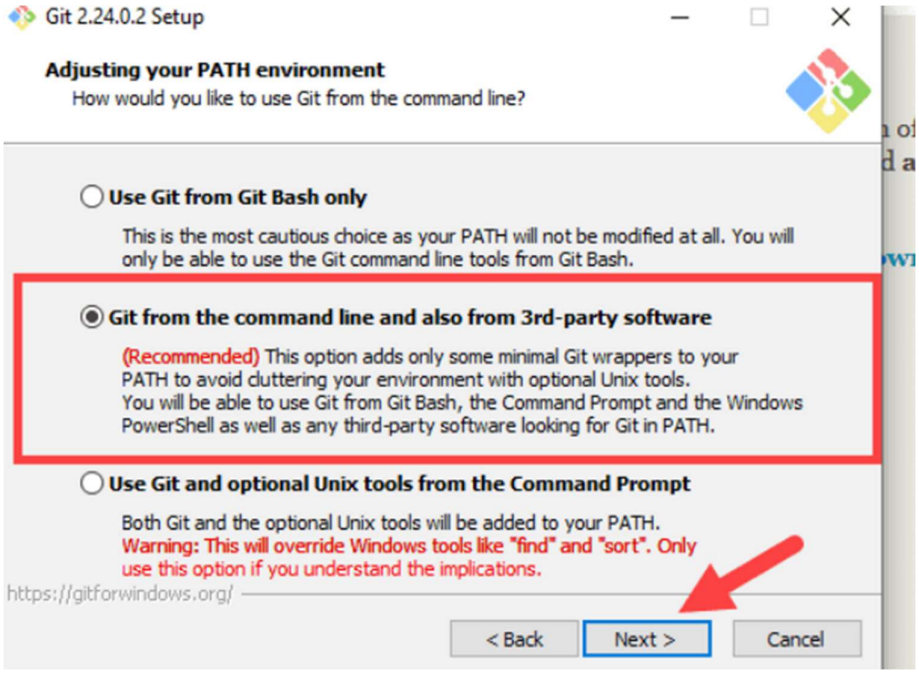
8. Select a text editor you'd like to use with Git. Use the drop-down menu to select Notepad++ (or whichever text editor you prefer) and click **Next**.



9. The next step allows you to choose a different name for your initial branch. The default is 'master.' Unless you're working in a team that requires a different name, leave the default option and click **Next.**



10. This installation step allows you to change the **PATH environment**. The **PATH** is the default set of directories included when you run a command from the command line. Leave this on the middle (recommended) selection and click **Next**.



11. The installer now asks which SSH client you want Git to use. Git already comes with its own SSH client, so if you don't need a specific one, leave the default option and click **Next.**

12. The next option relates to server certificates. Most users should use the default. If you're working in an Active Directory environment, you may need to switch to Windows Store certificates. Click **Next**.

13. The next selection converts line endings. It is recommended that you leave the default selection. This relates to the way data is formatted and changing this option may cause problems. Click **Next**.

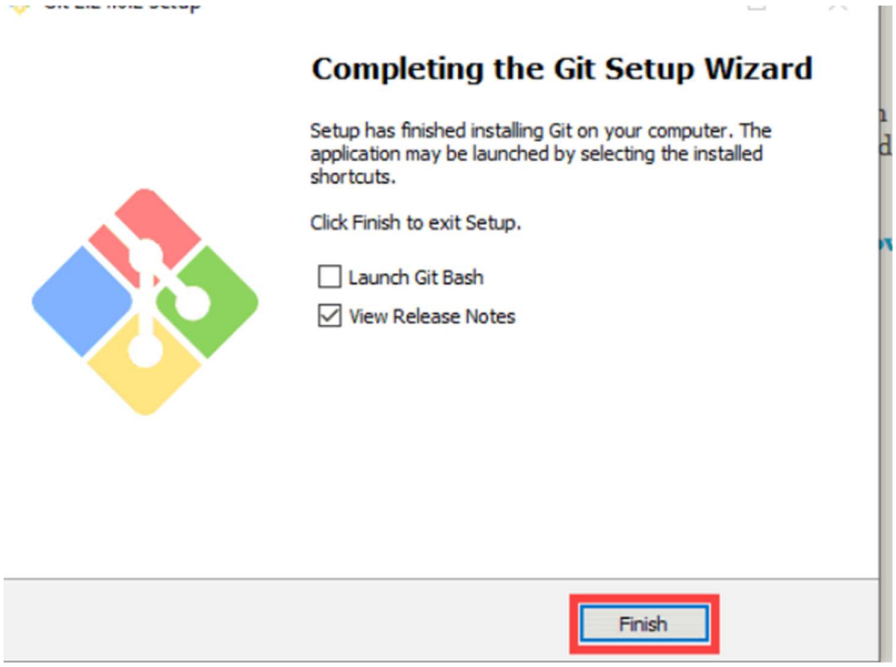
14. Choose the terminal emulator you want to use. The default MinTTY is recommended, for its features. Click **Next**.

15. Next you should choose which credential helper to use. Git uses credential helpers to fetch or save credentials. Leave the default option as it is the most stable one, and click **Next**.

16. The default options are recommended, however this step allows you to decide which extra option you would like to enable. If you use symbolic links, which are like shortcuts for the command line, tick the box. Click **Next**.

17. Depending on the version of Git you're installing, it may offer to install experimental features. Unless you are feeling adventurous, leave them unchecked and click **Install**.

18. Once the installation is complete, tick the boxes to view the Release Notes or Launch Git Bash, then click **Finish**.

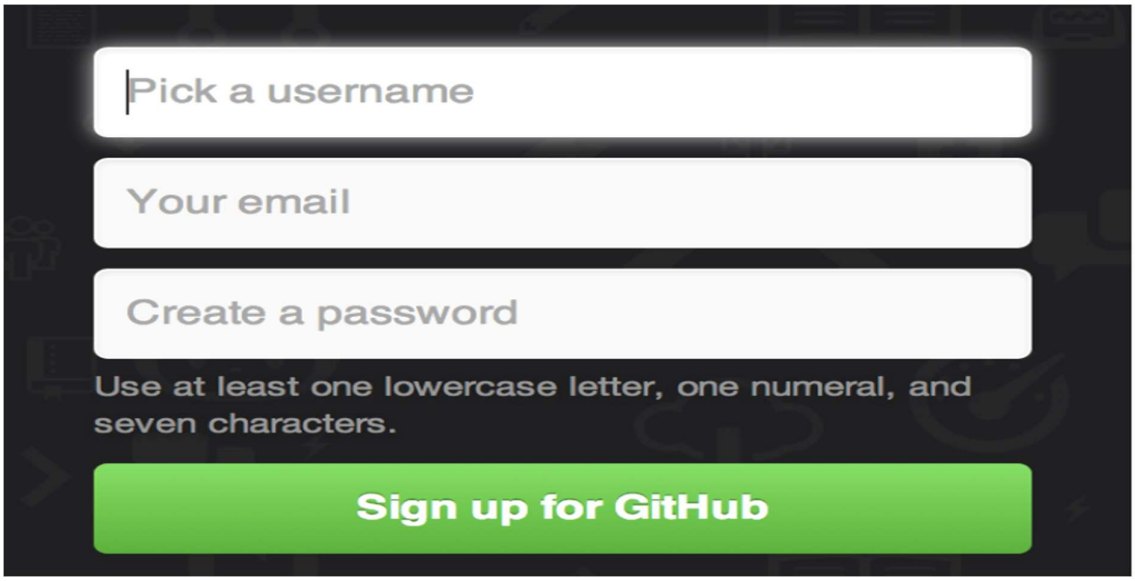


1) SETTING UP GIT HUB ACCOUNT:

GitHub is the single largest host for Git repositories, and is the central point of collaboration for millions of developers and projects. A large percentage of all Git repositories are hosted on GitHub, and many open-source projects use it for Git hosting, issue tracking, code review, and other things. So while it's not a direct part of the Git open source project, there's a good chance that you'll want or need to interact with GitHub at some point while using Git professionally.

This chapter is about using GitHub effectively. We'll cover signing up for and managing an account, creating and using Git repositories, common workflows to contribute to projects and to accept contributions to yours, GitHub's programmatic interface and lots of little tips to make your life easier in general.

The first thing you need to do is set up a free user account. Simply visit https://github.com, choose a user name that isn't already taken, provide an email address and a password, and click the big green "Sign up for GitHub" button.

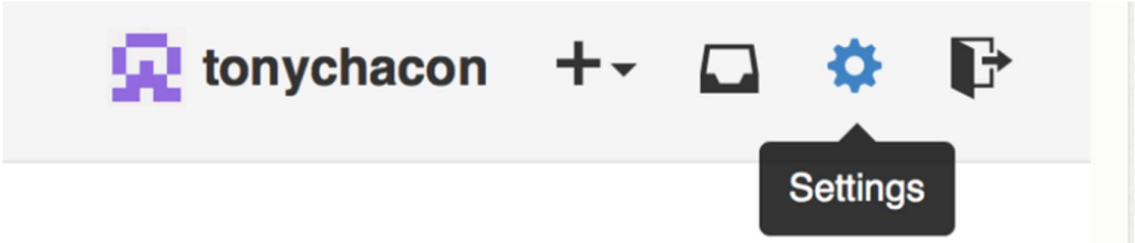


The next thing you'll see is the pricing page for upgraded plans, but it's safe to ignore this for now. GitHub will send you an email to verify the address you provided. Go ahead and do this; it's pretty important.

Clicking the Octocat logo at the top-left of the screen will take you to your dashboard page. You're now ready to use GitHub.

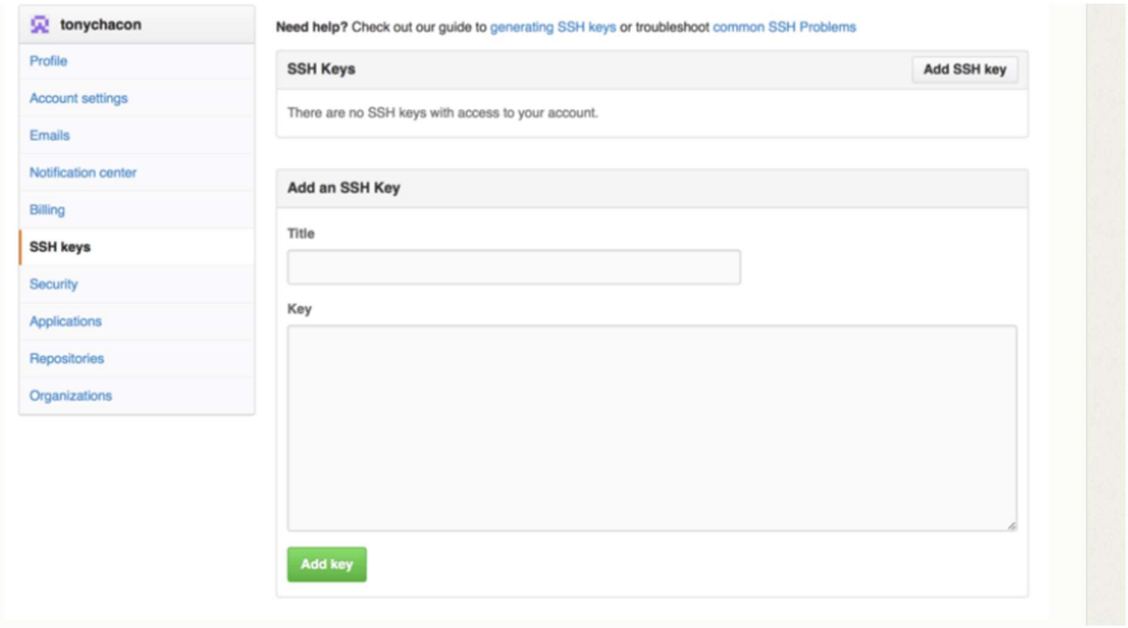
As of right now, you're fully able to connect with Git repositories using the `https://` protocol, authenticating with the username and password you just set up. However, to simply clone public projects, you don't even need to sign up - the account we just created comes into play when we fork projects and push to our forks a bit later.

If you'd like to use SSH remotes, you'll need to configure a public key. If you don't already have one, see Generating Your SSH Public Key. Open up your account settings using the link at the top-right of the window:



The "Account settings" link

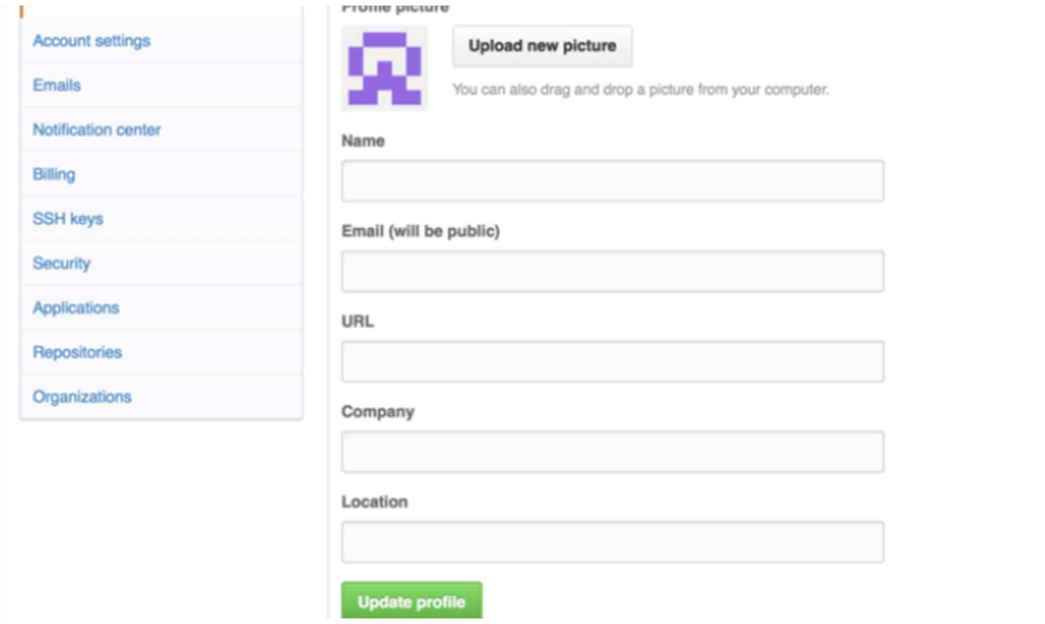
Then select the "SSH keys" section along the left-hand side.



The “SSH keys” link.

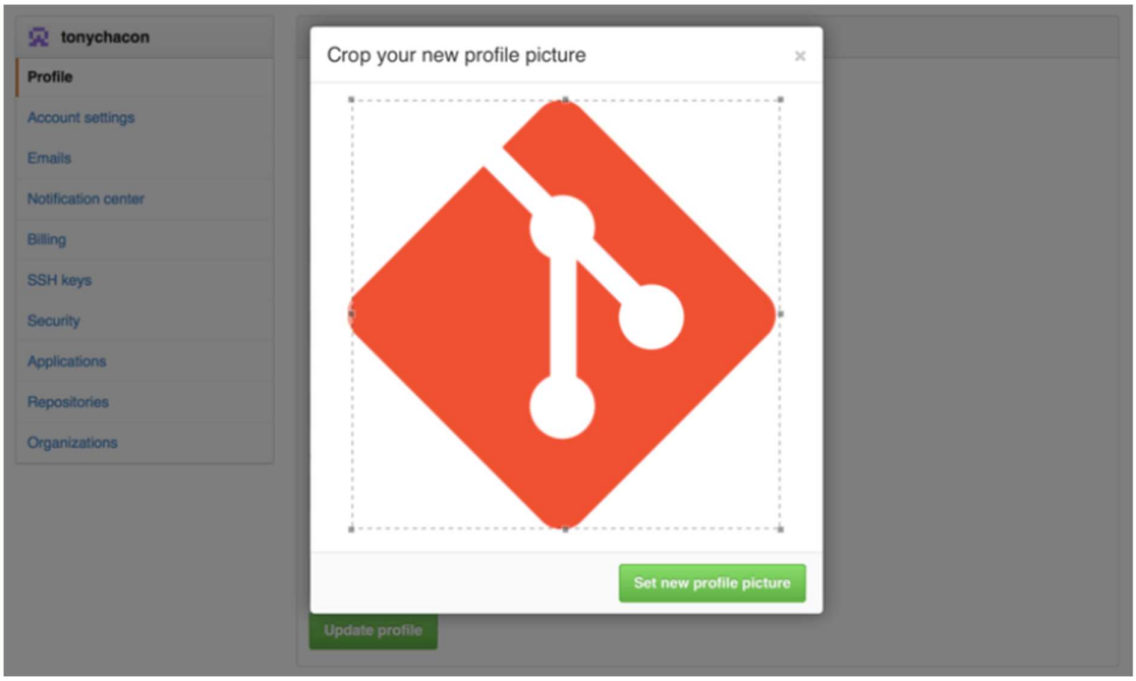
From there, click the “Add an SSH key” button, give your key a name, paste the contents of your `~/.ssh/id_rsa.pub` (or whatever you named it) public-key file into the text area, and click “Add key”.

Next, if you wish, you can replace the avatar that is generated for you with an image of your choosing. First go to the “Profile” tab (above the SSH Keys tab) and click “Upload new picture”.



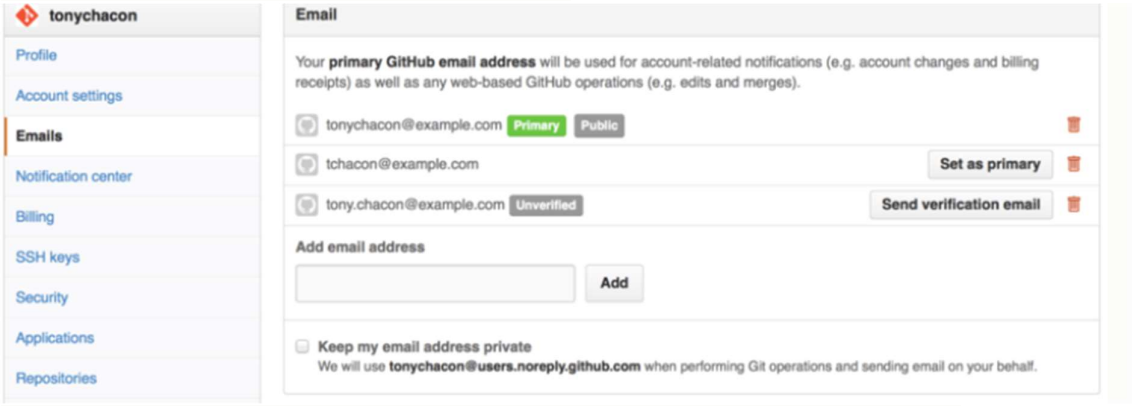
The “Profile” link

We’ll choose a copy of the Git logo that is on our hard drive and then we get a chance to crop it.



Now anywhere you interact on the site, people will see your avatar next to your username.

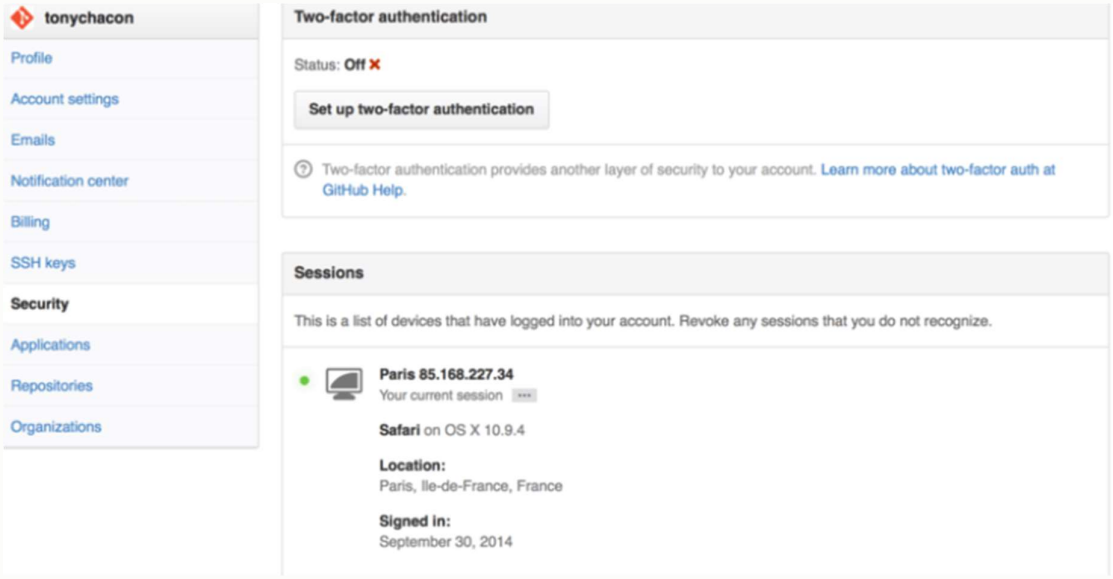
The way that GitHub maps your Git commits to your user is by email address. If you use multiple email addresses in your commits and you want GitHub to link them up properly, you need to add all the email addresses you have used to the Emails section of the admin section.



In Add email addresses we can see some of the different states that are possible. The top address is verified and set as the primary address, meaning that is where you'll get any notifications and receipts. The second address is verified and so can be set as the primary if you wish to switch them. The final address is unverified, meaning that you can't make it your primary address. If GitHub sees any of these in commit messages in any repository on the site, it will be linked to your user now.

Two Factor Authentication

Finally, for extra security, you should definitely set up Two-factor Authentication or “2FA”. Two-factor Authentication is an authentication mechanism that is becoming more and more popular recently to mitigate the risk of your account being compromised if your password is stolen somehow. Turning it on will make GitHub ask you for two different methods of authentication, so that if one of them is compromised, an attacker will not be able to access your account.



If you click on the “Set up two-factor authentication” button, it will take you to a configuration page where you can choose to use a phone app to generate your secondary code (a “time based one-time password”), or you can have GitHub send you a code via SMS each time you need to log in.

After you choose which method you prefer and follow the instructions for setting up 2FA, your account will then be a little more secure and you will have to provide a code in addition to your password whenever you log into GitHub.

2) Generate Log:

Used the following commands in local folder:

```
337  git init
338  git status
339  git add .
340  git status
341  git status
342  git commit -m "deleted an extra file"
343  git log
344  git status
345  git commit -m "deleted 2 extra files and modified 1 existing file"
346  git log
347  git add .
348  git log
349  git status
350  git commit -m" commit 2 and 3 deleted and modified"
351  git log
```

```
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        deleted:    PYTHAGORIAN TRIPLETUSING FUNCTION.exe
        deleted:    linear search.exe
        modified:   sum of n natural nosusing function.cpp
        deleted:    sum of n natural nosusing function.exe

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:   hollow rectangleof stars.cpp


devan@DESKTOP-E77QH23 MINGW64 ~/OneDrive/Desktop/patterns codes (master)
$ git commit -m" commit 2 and 3 deleted and modified"
[master 3a01cf7]  commit 2 and 3 deleted and modified
 4 files changed, 1 insertion(+)
 delete mode 100644 PYTHAGORIAN TRIPLETUSING FUNCTION.exe
 delete mode 100644 linear search.exe
 delete mode 100644 sum of n natural nosusing function.exe

devan@DESKTOP-E77QH23 MINGW64 ~/OneDrive/Desktop/patterns codes (master)
$ git log
commit 3a01cf773a362b42cfdb1f820d83d7959af0b3bf (HEAD -> master)
Author: DEVANSH-DOGRA <devansh0422.be21@chitkara.edu.in>
Date:   Sun Apr 10 11:10:41 2022 +0530

    commit 2 and 3 deleted and modified

commit 6a78e9d8f251e8c0dcf1eb5f48246971643ec0f7
Author: DEVANSH-DOGRA <devansh0422.be21@chitkara.edu.in>
Date:   Sun Apr 10 11:06:54 2022 +0530

    deleted an extra file
```

git init: The git init command creates a new Git repository. It can be used to convert an existing, unversioned project to a Git repository or initialize a new, empty repository. Most other Git commands are not available outside of an initialized repository, so this is usually the first command you'll run in a new project.

git status: The git status command is used to display the state of the repository and staging area. It allows us to see the tracked, untracked files and changes. This command will not show any commit records or information. Mostly, it is used to display the state between Git Add and Git commit command.

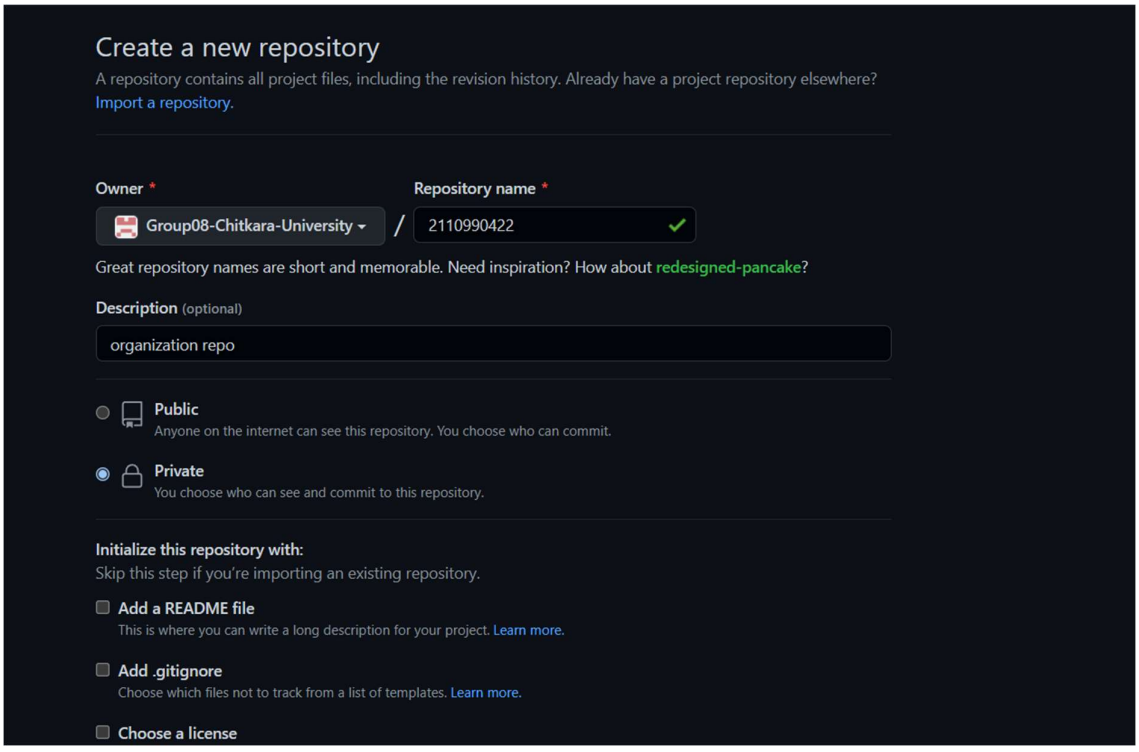
git add: The git add command adds a file to the Git staging area. This area contains a list of all the files you have recently changed. Your repository will be updated the next time you create a commit with your changes.

Therefore, running the git add command does not change any of your work in the Git repository. Changes are only made to your repository when you execute the git commit command.

git commit: The git commit command is one of the core primary functions of Git. Prior use of the git add command is required to select the changes that will be staged for the next commit. Then git commit is used to create a snapshot of the staged changes along a timeline of a Git projects history.

git log: The git log command shows a list of all the commits made to a repository. You can see the hash of each Git commit , the message associated with each commit, and more metadata. This command is useful for displaying the history of a repository.

3) Creating and visualising branches:



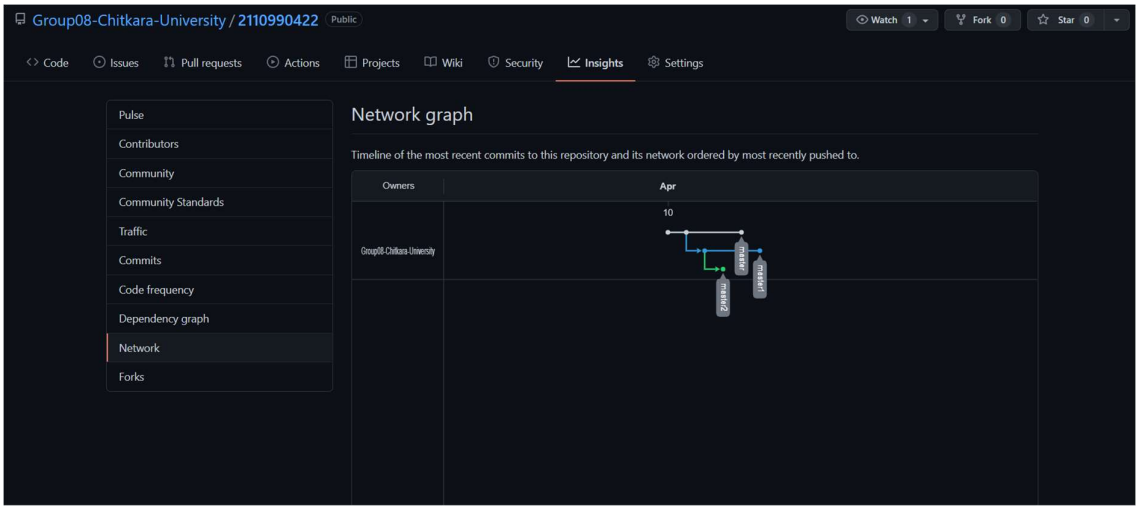
```
devan@DESKTOP-E77QH23 MINGW64 ~/OneDrive/Desktop/patterns codes (master)
$ git remote add origin https://github.com/Group08-Chitkara-University/2110990422.git

devan@DESKTOP-E77QH23 MINGW64 ~/OneDrive/Desktop/patterns codes (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:   hollow rectangleof stars.cpp

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

devan@DESKTOP-E77QH23 MINGW64 ~/OneDrive/Desktop/patterns codes (master)
$ git push -u origin master
Enumerating objects: 29, done.
Counting objects: 100% (29/29), done.
Delta compression using up to 8 threads
Compressing objects: 100% (29/29), done.
Writing objects: 100% (29/29), 67.37 KiB | 1.73 MiB/s, done.
Total 29 (delta 10), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (10/10), done.
To https://github.com/Group08-Chitkara-University/2110990422.git
 * [new branch]      master -> master
branch 'master' set up to track 'origin/master'.
```

```
devan@DESKTOP-E77QH23 MINGW64 ~/OneDrive/Desktop/patterns codes (master)
$ git branch master1

devan@DESKTOP-E77QH23 MINGW64 ~/OneDrive/Desktop/patterns codes (master)
$ git checkout master1
Switched to branch 'master1'
M       hollow rectangleof stars.cpp

devan@DESKTOP-E77QH23 MINGW64 ~/OneDrive/Desktop/patterns codes (master1)
$ git branch
  master
* master1

devan@DESKTOP-E77QH23 MINGW64 ~/OneDrive/Desktop/patterns codes (master1)
$ git status
On branch master1
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:   hollow rectangleof stars.cpp

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

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

devan@DESKTOP-E77QH23 MINGW64 ~/OneDrive/Desktop/patterns codes (master1)
$ git commit -m "changes to new branch"
On branch master1
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:   hollow rectangleof stars.cpp

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

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

devan@DESKTOP-E77QH23 MINGW64 ~/OneDrive/Desktop/patterns codes (master1)
$ git add .

devan@DESKTOP-E77QH23 MINGW64 ~/OneDrive/Desktop/patterns codes (master1)
$ git commit -m "changes to new branch"
[master1 804d545] changes to new branch
 2 files changed, 14 insertions(+)
 create mode 100644 file names.txt

devan@DESKTOP-E77QH23 MINGW64 ~/OneDrive/Desktop/patterns codes (master1)
Everything up-to-datemaster
branch 'master' set up to track 'origin/master'.

devan@DESKTOP-E77QH23 MINGW64 ~/OneDrive/Desktop/patterns codes (master1)
$ git push -u origin master1
```

```
devan@DESKTOP-E77QH23 MINGW64 ~/OneDrive/Desktop/patterns codes (master1)
$ git branch master2

devan@DESKTOP-E77QH23 MINGW64 ~/OneDrive/Desktop/patterns codes (master1)
$ git branch
  master
* master1
  master2

devan@DESKTOP-E77QH23 MINGW64 ~/OneDrive/Desktop/patterns codes (master1)
$ git checkout master2
Switched to branch 'master2'

devan@DESKTOP-E77QH23 MINGW64 ~/OneDrive/Desktop/patterns codes (master2)
$ git branch
  master
  master1
* master2

devan@DESKTOP-E77QH23 MINGW64 ~/OneDrive/Desktop/patterns codes (master2)
$ git add .

devan@DESKTOP-E77QH23 MINGW64 ~/OneDrive/Desktop/patterns codes (master2)
$ git commit -m "commit for master 3 pascal triangle"
[master2 b38bb64] commit for master 3 pascal triangle
 1 file changed, 45 insertions(+)
 create mode 100644 codeee.txt
```

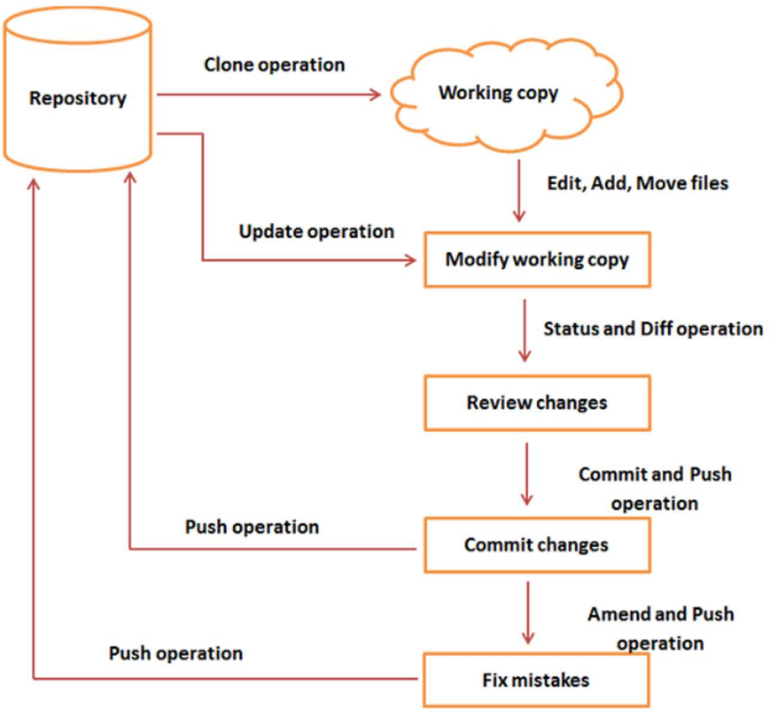
```
devan@DESKTOP-E77QH23 MINGW64 ~/OneDrive/Desktop/patterns codes (master2)
$ git push -u origin master2
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 630 bytes | 630.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
remote:
remote: Create a pull request for 'master2' on GitHub by visiting:
remote:      https://github.com/Group08-Chitkara-University/2110990422/pull/new/master2
remote:
To https://github.com/Group08-Chitkara-University/2110990422.git
 * [new branch]      master2 -> master2
branch 'master2' set up to track 'origin/master2'.

devan@DESKTOP-E77QH23 MINGW64 ~/OneDrive/Desktop/patterns codes (master2)
$ git checkout master
Switched to branch 'master'
Your branch is up to date with 'origin/master'.

devan@DESKTOP-E77QH23 MINGW64 ~/OneDrive/Desktop/patterns codes (master)
$ git branch
* master
  master1
  master2

devan@DESKTOP-E77QH23 MINGW64 ~/OneDrive/Desktop/patterns codes (master)
$ git status
On branch master
Your branch is up to date with 'origin/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:   hollow rectangleof stars.cpp

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

devan@DESKTOP-E77QH23 MINGW64 ~/OneDrive/Desktop/patterns codes (master)
$ git add .

devan@DESKTOP-E77QH23 MINGW64 ~/OneDrive/Desktop/patterns codes (master)
$ git push -u origin master
Everything up-to-date
branch 'master' set up to track 'origin/master'.

devan@DESKTOP-E77QH23 MINGW64 ~/OneDrive/Desktop/patterns codes (master)
$ git checkout master1
error: Your local changes to the following files would be overwritten by checkout:
        hollow rectangleof stars.cpp
Please commit your changes or stash them before you switch branches.
Aborting

devan@DESKTOP-E77QH23 MINGW64 ~/OneDrive/Desktop/patterns codes (master)
$ git branch
```

```
devan@DESKTOP-E77QH23 MINGW64 ~/OneDrive/Desktop/patterns codes (master)
$ git branch
* master
  master1
  master2

devan@DESKTOP-E77QH23 MINGW64 ~/OneDrive/Desktop/patterns codes (master)
$ git commit -m "master changes"
[master 4b09025] master changes
 1 file changed, 18 insertions(+)

devan@DESKTOP-E77QH23 MINGW64 ~/OneDrive/Desktop/patterns codes (master)
$ git push -u origin master
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 320 bytes | 160.00 KiB/s, done.
Total 3 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/Group08-Chitkara-University/2110990422.git
   3a01cf7..4b09025  master -> master
branch 'master' set up to track 'origin/master'.

devan@DESKTOP-E77QH23 MINGW64 ~/OneDrive/Desktop/patterns codes (master)
$ git checkout master1
Switched to branch 'master1'
Your branch is up to date with 'origin/master1'.

devan@DESKTOP-E77QH23 MINGW64 ~/OneDrive/Desktop/patterns codes (master1)
$ git add .

devan@DESKTOP-E77QH23 MINGW64 ~/OneDrive/Desktop/patterns codes (master1)
$ git commit -m "master1 changes-deleted linear search oo"
[master1 167ed2c] master1 changes-deleted linear search oo
 1 file changed, 0 insertions(+), 0 deletions(-)
 delete mode 100644 linear searchusing functions.exe

devan@DESKTOP-E77QH23 MINGW64 ~/OneDrive/Desktop/patterns codes (master1)
$ git push -u origin master1
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 266 bytes | 266.00 KiB/s, done.
Total 2 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/Group08-Chitkara-University/2110990422.git
   804d545..167ed2c  master1 -> master1
branch 'master1' set up to track 'origin/master1'.
```

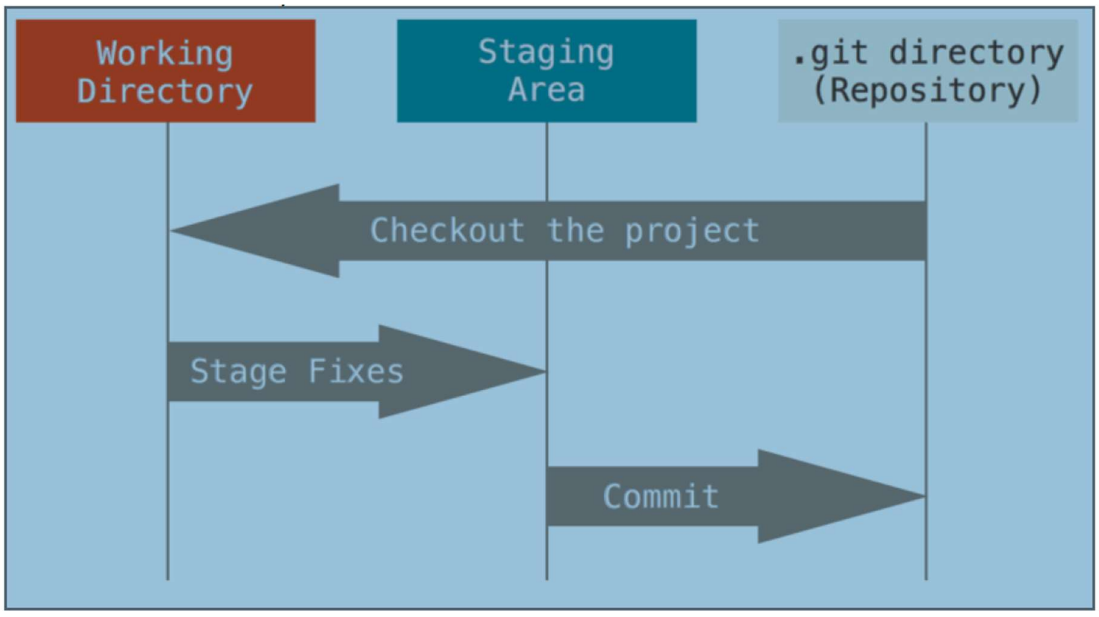


5)Git life cycle description:

Shown below is the pictorial representation of the work-flow:



When a directory is made a git repository, there are mainly 3 states which make the essence of Git version Control System. The three states are-



When a directory is made a git repository, there are mainly 3 states which make the essence of Git Version Control System. The three states are –

Working Directory
Staging Area
Git Directory

1. Working Directory
Whenever we want to initialize our local project directory to make it a git repository, we use the git init command. After this command, git becomes aware of the files in the project although it doesn't track the files yet. The files are further tracked in the staging area.

git init

2. Staging Area
Now, to track the different versions of our files we use the command git add. We can term a staging area as a place where different versions of our files are stored. git add command copies the version of your file from your working directory to the staging area. We can, however, choose which files we need to add to the staging area because in our working directory there are some files that we don't want to get tracked, examples include node modules, env files, temporary files, etc. Indexing in Git is the one that helps Git in understanding which files need to be added or sent. You can find your staging area in the .git folder inside the index file.

// to specify which file to add to the staging area

git add <filename>

// to add all files of the working directory to the staging area

git add .

3. Git Directory
Now since we have all the files that are to be tracked and are ready in the staging area, we are ready to commit our files using the git commit command. Commit helps us in keeping the track of the metadata of the files in our staging area. We specify every commit with a message which tells what the commit is about. Git preserves the information or the metadata of the files that were committed in a Git Directory which helps Git in tracking files and basically it preserves the photocopy of the committed files. Commit also stores the name of the author who did the commit, files that are committed, and the date at which they are committed along with the commit message.

git commit -m “Commit Message”

NAME: DEVANSH DOGRA
ROLL NO.- 2110990422
G08

TASK 1.2:

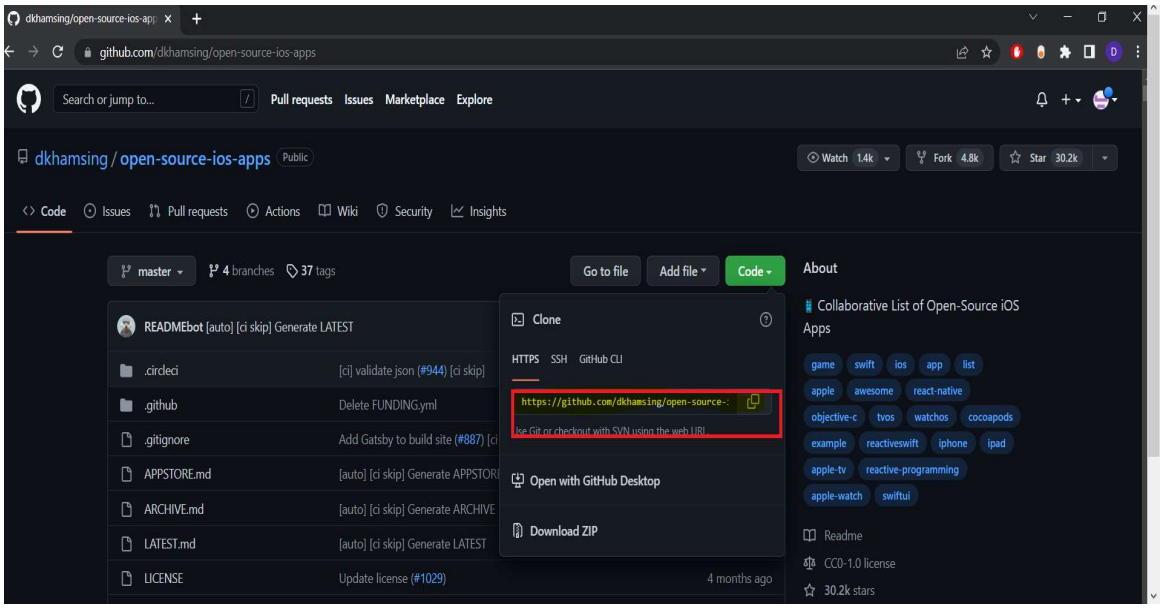
CLONING (git clone):

The git clone is a command-line utility which is used to make a local copy of a remote repository. It accesses the repository through a remote URL.

Now, here we are cloning an open source ios apps repository. On github we searched for open source ios apps and we got a collaborative list of really awesome ios apps.

There are actually two ways to clone a repository, either by directly downloading it as a ZIP file for which the option is available in that repository.

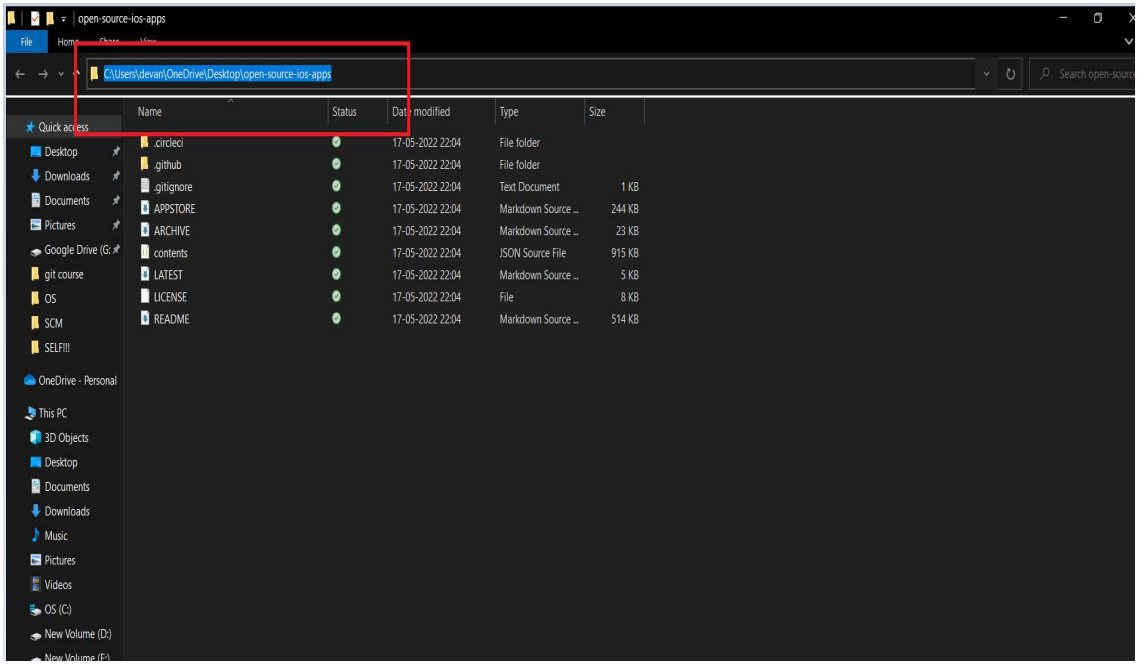
But we as command-line learners will always prefer cloning a repository using git bash. For cloning the repository git clone command is used.

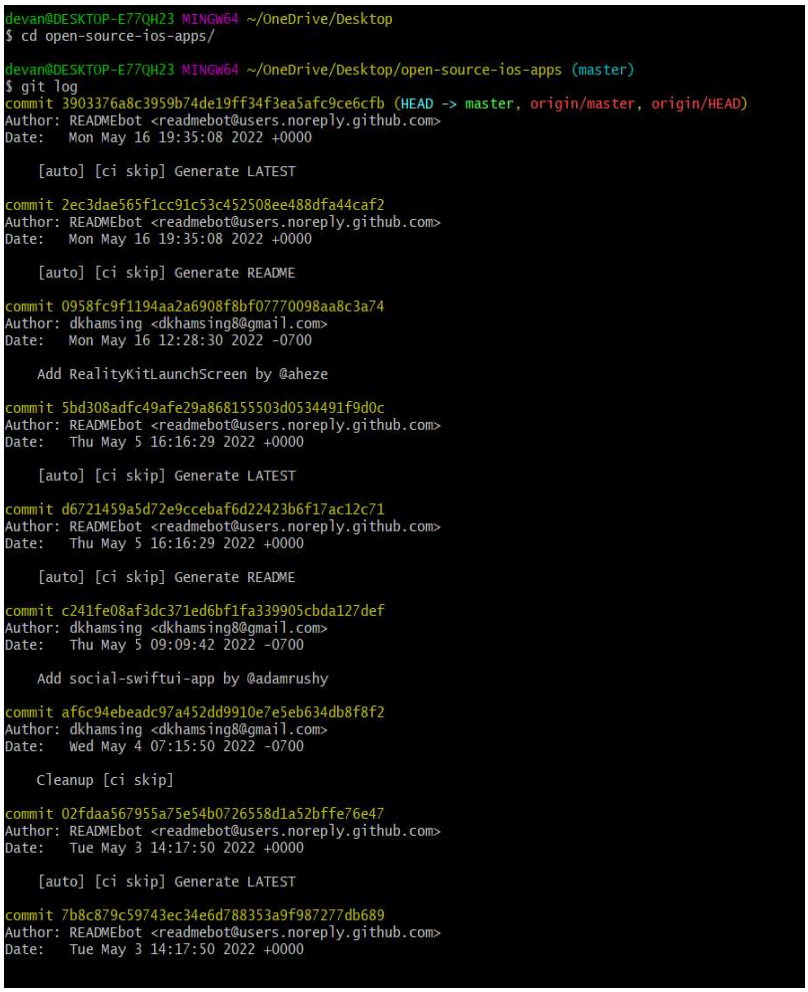


Syntax: git clone url of the repository that you want to clone

```
devan@DESKTOP-E77QH23 MINGW64 ~/OneDrive/Desktop
$ git clone https://github.com/dkhamsing/open-source-ios-apps.git
Cloning into 'open-source-ios-apps'...
remote: Enumerating objects: 17978, done.
remote: Counting objects: 100% (171/171), done.
remote: Compressing objects: 100% (65/65), done.
remote: Total 17978 (delta 110), reused 115 (delta 106), pack-reused 17807
Receiving objects: 100% (17978/17978), 16.86 MiB | 4.57 MiB/s, done.
Resolving deltas: 100% (11415/11415), done.
```

Now the open source ios app repository is successfully cloned.





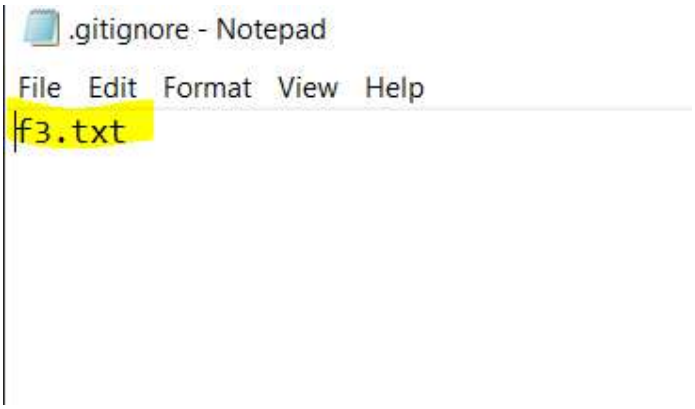
GIT IGNORE:

Git can specify which files or parts of your project should be ignored by Git using a .gitignore file. Git will not track files and folders specified in .gitignore

Here we have created a directory named gitignoredemo in which there are three files namely f1.txt, f2.txt and f3. txt. The file f3.txt is the file that we want to be ignored by git. So for this we created a new .gitignore file.

```
devan@DESKTOP-E77QH23 MINGW64 ~/OneDrive/Desktop
$ mkdir gitignoredemo

devan@DESKTOP-E77QH23 MINGW64 ~/OneDrive/Desktop
$ cd gitignoredemo/

devan@DESKTOP-E77QH23 MINGW64 ~/OneDrive/Desktop/gitignoredemo
$ touch f1.txt

devan@DESKTOP-E77QH23 MINGW64 ~/OneDrive/Desktop/gitignoredemo
$ touch f2.txt f3.txt

devan@DESKTOP-E77QH23 MINGW64 ~/OneDrive/Desktop/gitignoredemo
$ ls -a
./  ../  f1.txt  f2.txt  f3.txt

devan@DESKTOP-E77QH23 MINGW64 ~/OneDrive/Desktop/gitignoredemo
$ touch .gitignore

devan@DESKTOP-E77QH23 MINGW64 ~/OneDrive/Desktop/gitignoredemo
$ vim .gitignore
```

In this .gitignore file we type the names of those files that we want git to ignore.



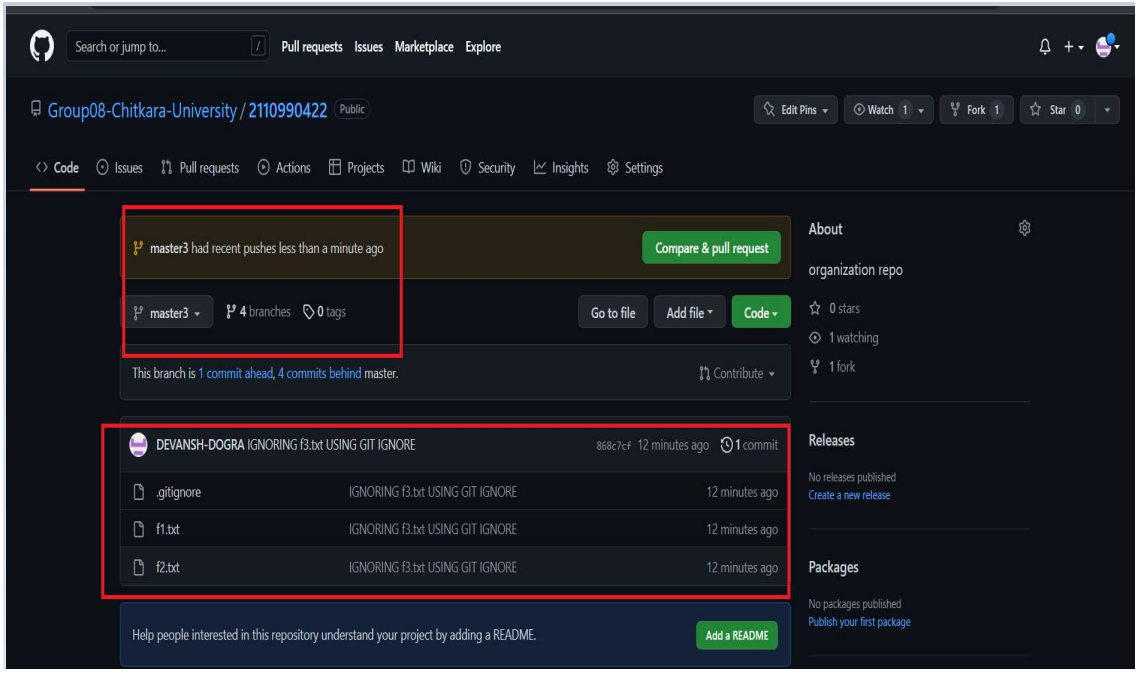
Then we follow the accustomed steps to push the files into the remote repository.

```
devan@DESKTOP-E77QH23 MINGW64 ~/OneDrive/Desktop/gitignoredemo
$ git init
Initialized empty Git repository in C:/Users/devan/OneDrive/Desktop/gitignoredemo/.git/

devan@DESKTOP-E77QH23 MINGW64 ~/OneDrive/Desktop/gitignoredemo (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .gitignore
        f1.txt
        f2.txt

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

devan@DESKTOP-E77QH23 MINGW64 ~/OneDrive/Desktop/gitignoredemo (master)
$ git add .
warning: LF will be replaced by CRLF in .gitignore.
The file will have its original line endings in your working directory

devan@DESKTOP-E77QH23 MINGW64 ~/OneDrive/Desktop/gitignoredemo (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   .gitignore
        new file:   f1.txt
        new file:   f2.txt

devan@DESKTOP-E77QH23 MINGW64 ~/OneDrive/Desktop/gitignoredemo (master)
$ git commit -m "IGNORING f3.txt USING GIT IGNORE"
[master (root-commit) 868c7cf] IGNORING f3.txt USING GIT IGNORE
 3 files changed, 2 insertions(+)
 create mode 100644 .gitignore
 create mode 100644 f1.txt
 create mode 100644 f2.txt
```

```
devan@DESKTOP-E77QH23 MINGW64 ~/OneDrive/Desktop/gitignoredemo (master)
$ git remote add origin https://github.com/Group08-Chitkara-University/2110990422.git
```

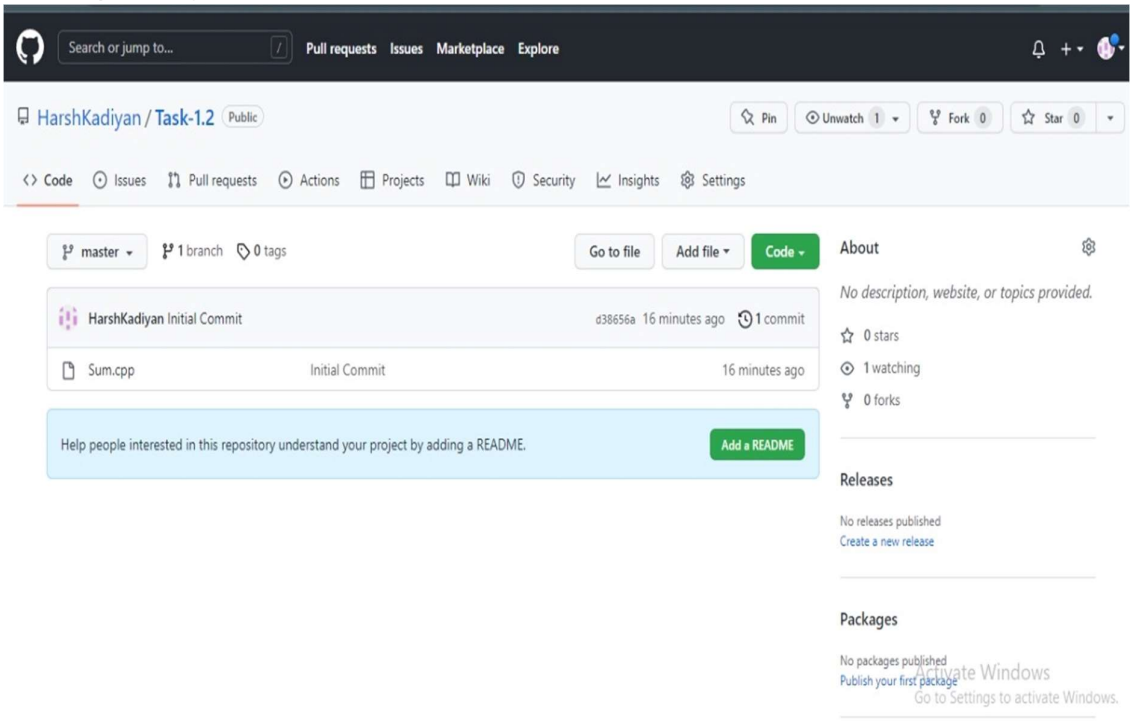
```
devan@DESKTOP-E77QH23 MINGW64 ~/OneDrive/Desktop/gitignoredemo (master)
$ git branch
* master

devan@DESKTOP-E77QH23 MINGW64 ~/OneDrive/Desktop/gitignoredemo (master)
$ git branch master3

devan@DESKTOP-E77QH23 MINGW64 ~/OneDrive/Desktop/gitignoredemo (master)
$ git branch
* master
  master3

devan@DESKTOP-E77QH23 MINGW64 ~/OneDrive/Desktop/gitignoredemo (master)
$ git checkout master3
Switched to branch 'master3'

devan@DESKTOP-E77QH23 MINGW64 ~/OneDrive/Desktop/gitignoredemo (master3)
$ git push -u origin master3
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% (4/4), 301 bytes | 301.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'master3' on GitHub by visiting:
remote:      https://github.com/Group08-Chitkara-University/2110990422/pull/new/master3
remote:
To https://github.com/Group08-Chitkara-University/2110990422.git
 * [new branch]      master3 -> master3
branch 'master3' set up to track 'origin/master3'.
```

Now we have successfully pushed the desired files(f1.txt and f2.txt) and ignoring the specific file(f3.txt) on our remote repository on github.

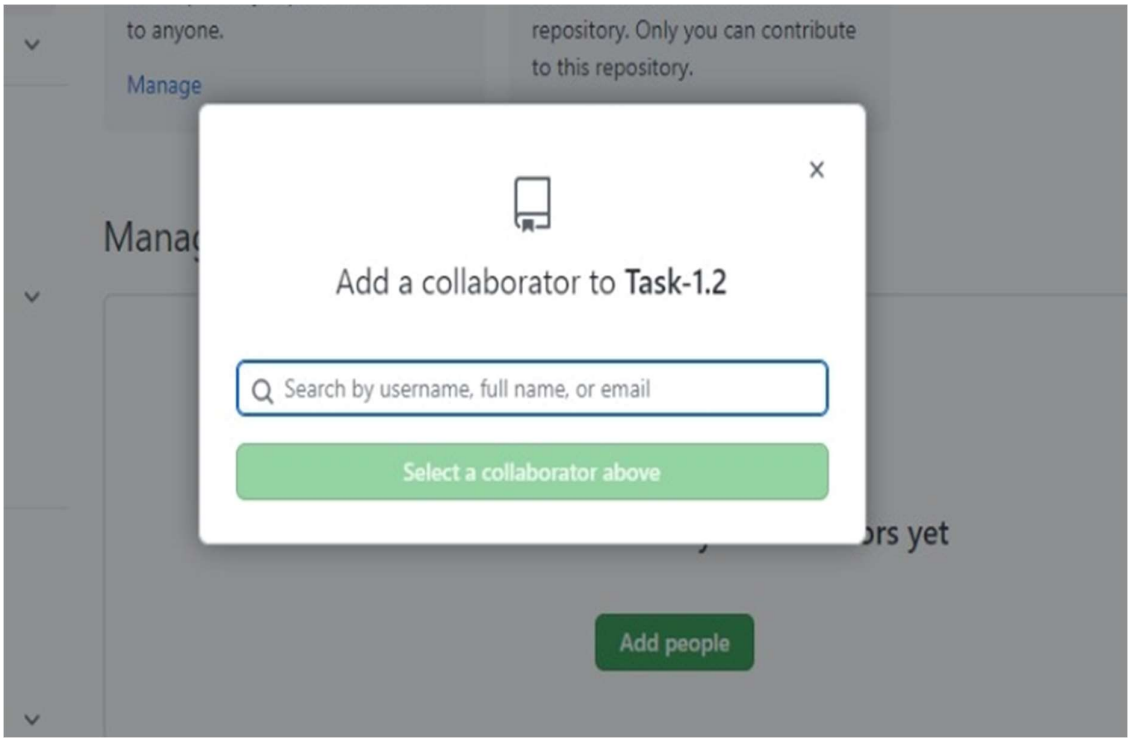


MERGING AND RESOLVING CONFLICTS IN GIT:

New repository created:



Adding collaborators on the new repository created:



Manage

Manage access

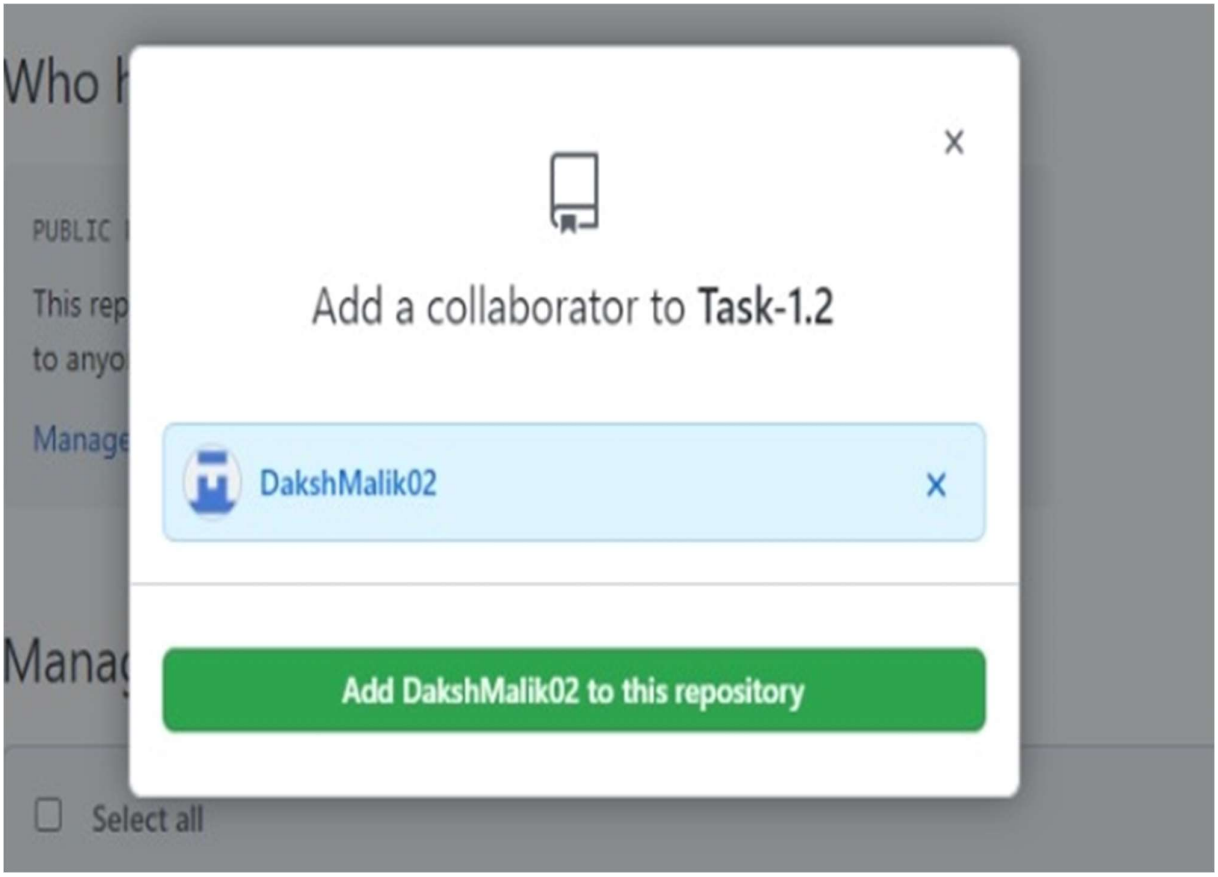
Add people

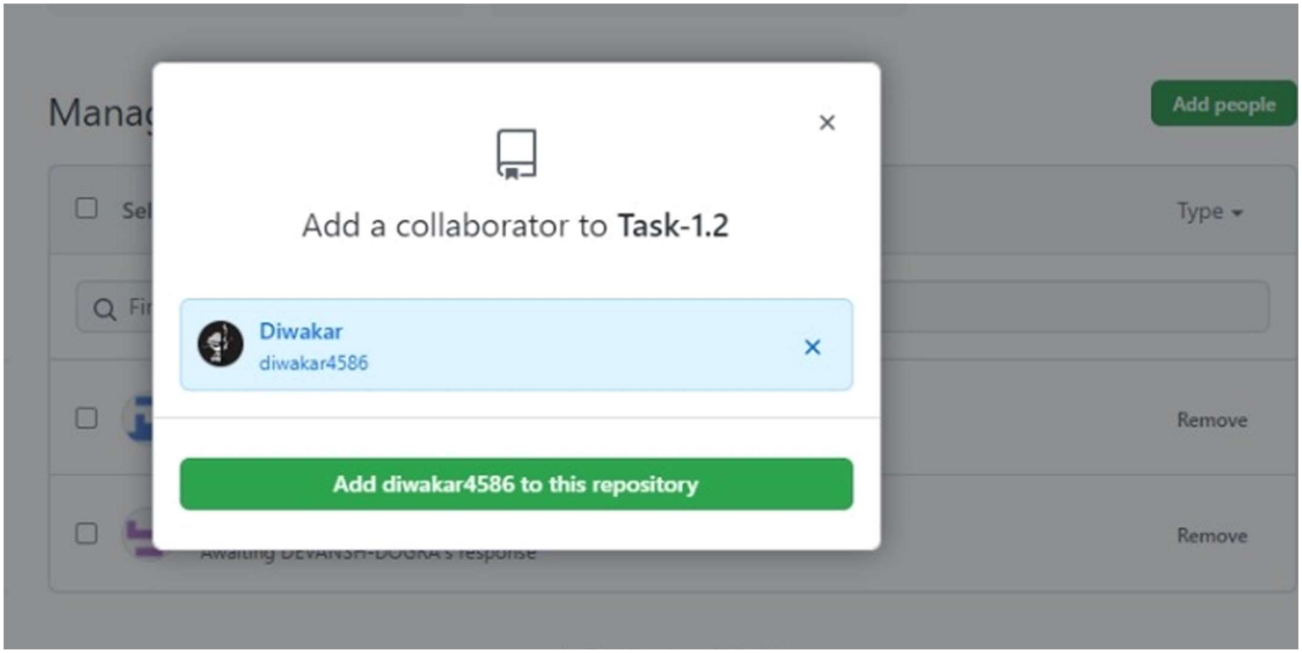
Select all Type

Find a collaborator...

DEVANSH-DOGRA
Awaiting DEVANSH-DOGRA's response
Pending Invite
Remove

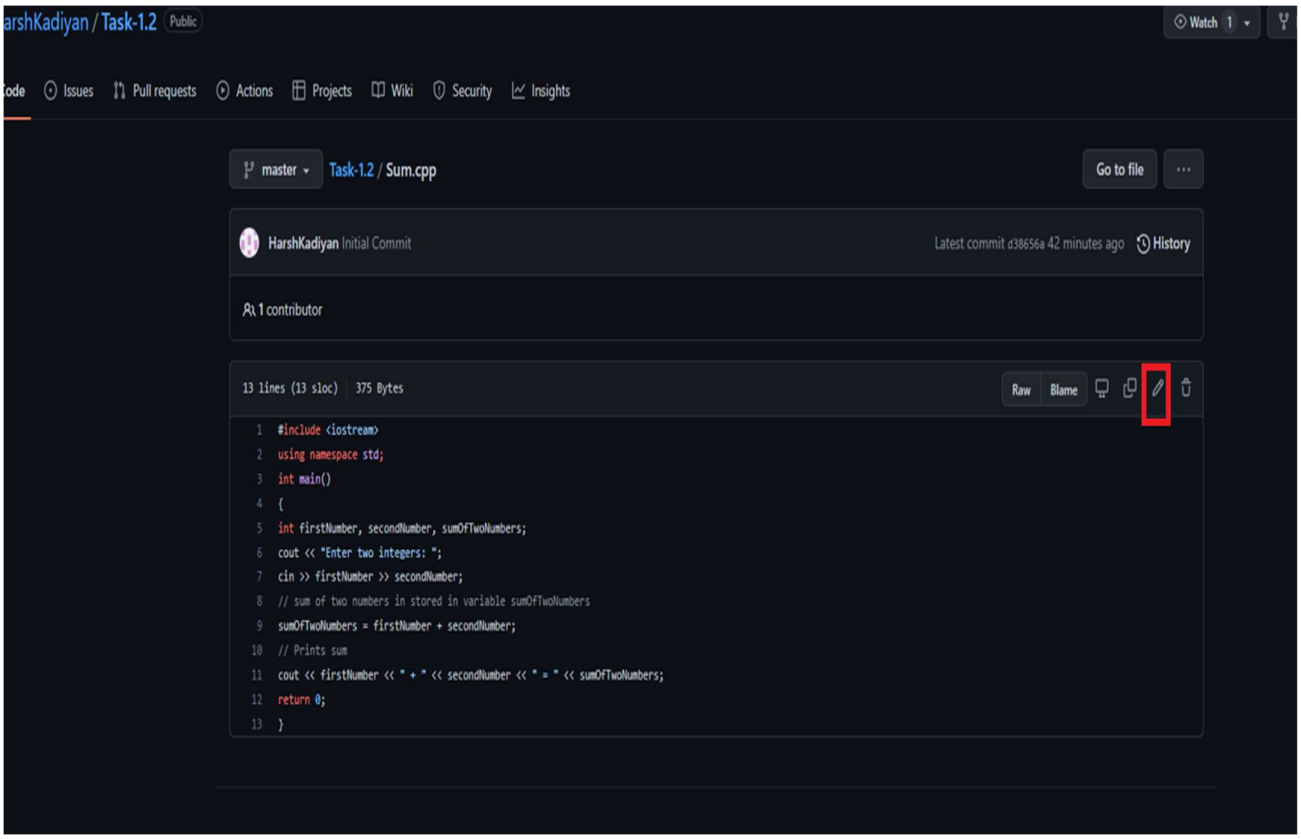
‹ Previous Next ›





Making changes to the file (collaborators activity), that will lead to conflicts:

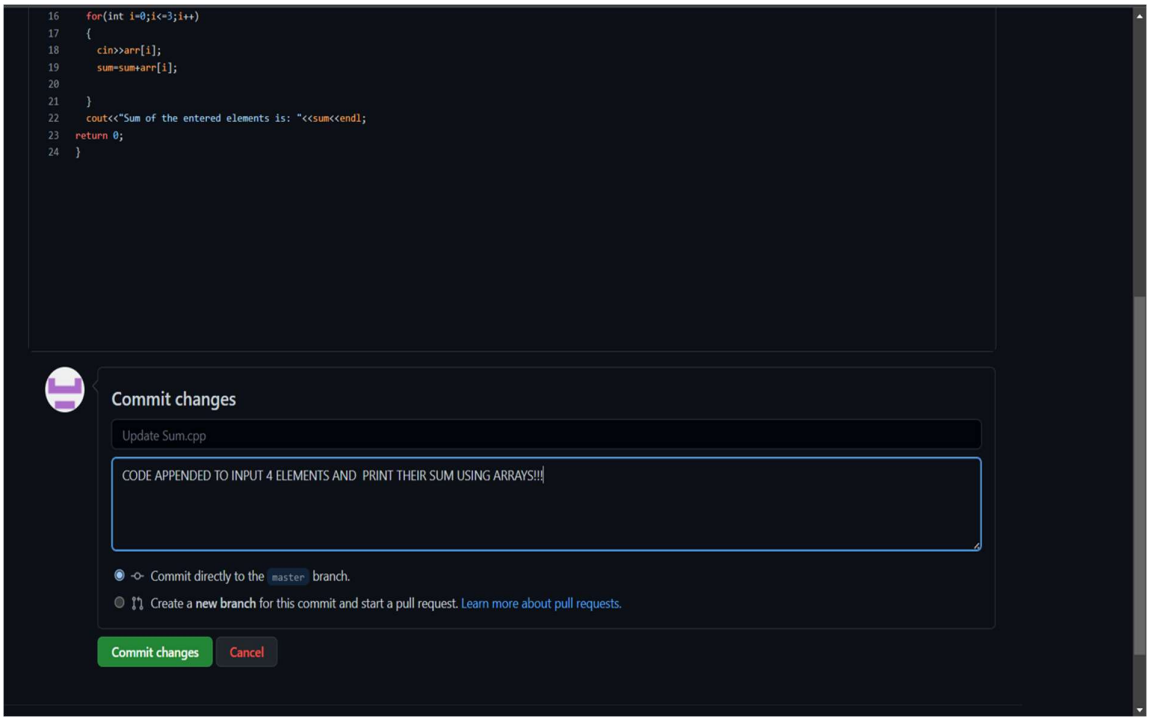
Making changes in the file (online editor) on gut hub:



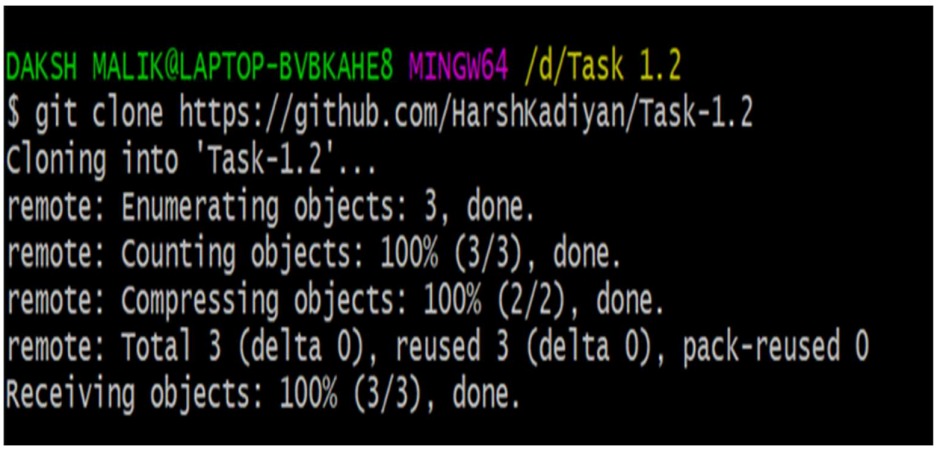
Changes made:

24 lines (23 sloc) | 696 Bytes

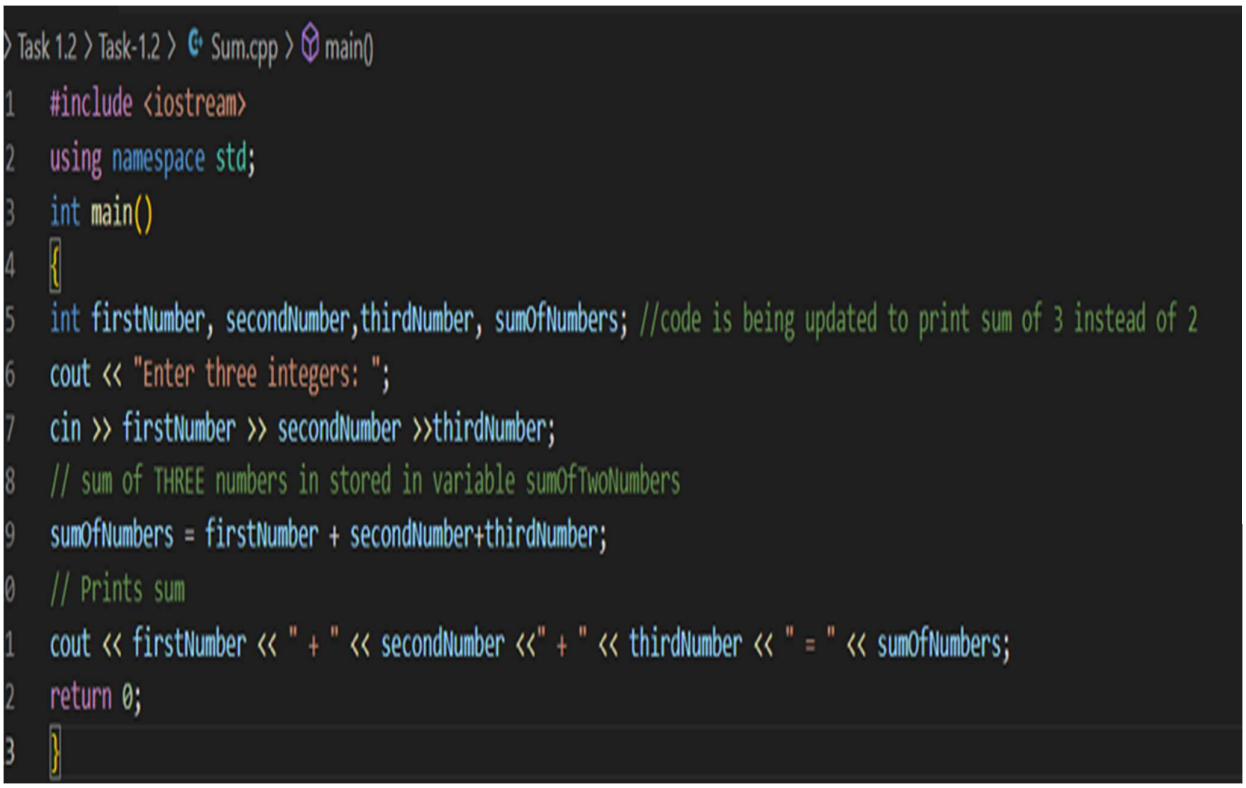
```cpp
#include <iostream>
using namespace std;
int main()
{
int firstNumber, secondNumber, sumOfTwoNumbers;
cout << "Enter two integers: ";
cin >> firstNumber >> secondNumber;
// sum of two numbers in stored in variable sumOfTwoNumbers
sumOfTwoNumbers = firstNumber + secondNumber;
// Prints sum
cout << firstNumber << " + " << secondNumber << " = " << sumOfTwoNumbers<< endl;
 //I AM DEVANSH. I AM APPENDING A CODE THAT TAKES 4 ARRAY ELEMENTS AS INPUT AND THEN PRINTS THEIR SUM !!!!!!!!
 int arr[4];
 int sum=0;
 cout<<"Enter 4 elements to sum up:"<<endl;
 for(int i=0;i<=3;i++)
 {
   cin>>arr[i];
   sum=sum+arr[i];

 }
 cout<<"Sum of the entered elements is: "<<sum<<endl;
return 0;
}
```



Cloning the repository :



Editing the code locally:



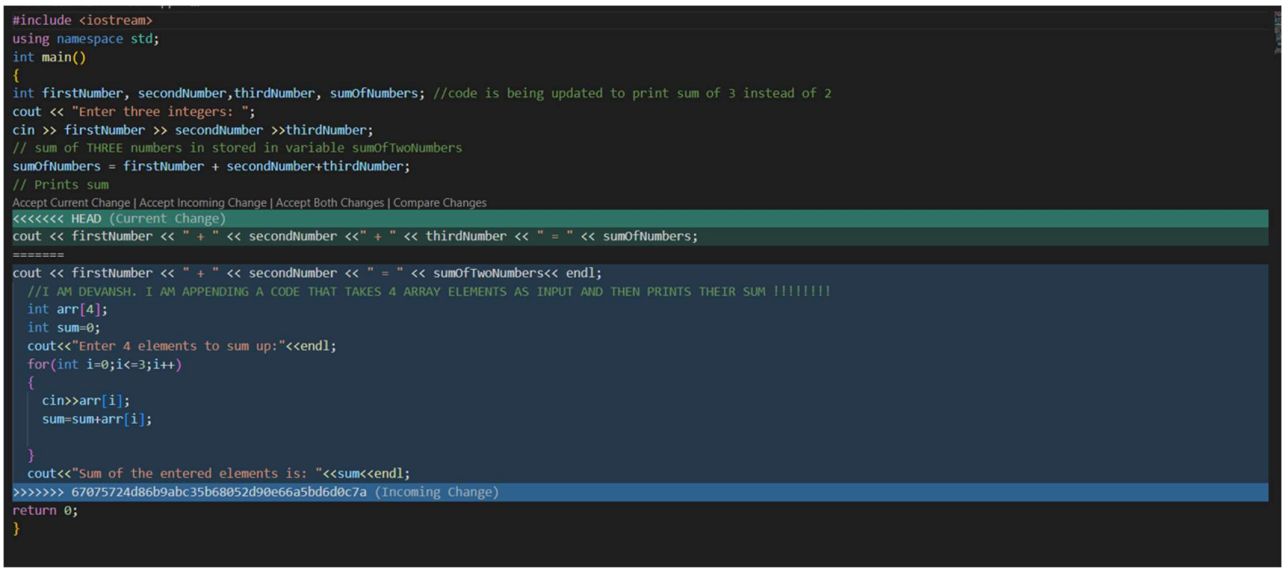
Conflict occurs while pushing the code:

```
DAKSH MALIK@LAPTOP-BVBKAHE8 MINGW64 /d/Task 1.2/Task-1.2 (master)
$ git status
On branch master
Your branch is up to date with 'origin/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:   Sum.cpp

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

DAKSH MALIK@LAPTOP-BVBKAHE8 MINGW64 /d/Task 1.2/Task-1.2 (master)
$ git add .

DAKSH MALIK@LAPTOP-BVBKAHE8 MINGW64 /d/Task 1.2/Task-1.2 (master)
$ git commit -m "Updated code"
[master 4435fc7] Updated code
 1 file changed, 13 insertions(+), 13 deletions(-)
 rewrite Sum.cpp (79%)

DAKSH MALIK@LAPTOP-BVBKAHE8 MINGW64 /d/Task 1.2/Task-1.2 (master)
$ git push -u origin master
To https://github.com/HarshKadiyan/Task-1.2
 ! [rejected]        master -> master (fetch first)
error: failed to push some refs to 'https://github.com/HarshKadiyan/Task-1.2'
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.
```

Now to resolve it, we will pull the original code:

```
DAKSH MALIK@LAPTOP-BVBKAHE8 MINGW64 /d/Task 1.2/Task-1.2 (master)
$ git pull
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), 1.03 KiB | 151.00 KiB/s, done.
From https://github.com/HarshKadiyan/Task-1.2
   d38656a..6707572  master     -> origin/master
Auto-merging Sum.cpp
CONFLICT (content): Merge conflict in Sum.cpp
Automatic merge failed; fix conflicts and then commit the result.
```

Further by opening the file with editor it shows changes made by the remote user:



```
#include <iostream>
using namespace std;
int main()
{
int firstNumber, secondNumber,thirdNumber, sumOfNumbers; //code is being updated to print sum of 3 instead of 2
cout << "Enter three integers: ";
cin >> firstNumber >> secondNumber >>thirdNumber;
// sum of THREE numbers in stored in variable sumOfTwoNumbers
sumOfNumbers = firstNumber + secondNumber+thirdNumber;
// Prints sum
Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes
<<<<<<< HEAD (Current Change)
cout << firstNumber << " + " << secondNumber <<" + " << thirdNumber << " = " << sumOfNumbers;
=======
cout << firstNumber << " + " << secondNumber << " = " << sumOfTwoNumbers<< endl;
  //I AM DEVANSH. I AM APPENDING A CODE THAT TAKES 4 ARRAY ELEMENTS AS INPUT AND THEN PRINTS THEIR SUM !!!!!!!!
  int arr[4];
  int sum=0;
  cout<<"Enter 4 elements to sum up:"<<endl;
  for(int i=0;i<=3;i++)
  {
    cin>>arr[i];
    sum=sum+arr[i];

  }
  cout<<"Sum of the entered elements is: "<<sum<<endl;
>>>>>>> 67075724d86b9abc35b68052d90e66a5bd6d0c7a (Incoming Change)
return 0;
}
```

Code is updated:

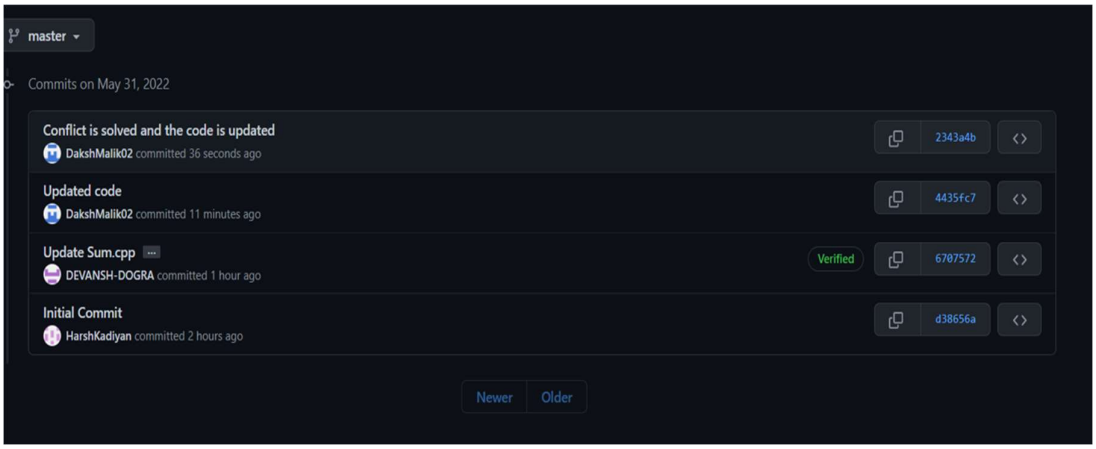
```
#include <iostream>
using namespace std;
int main()
{
int firstNumber, secondNumber,thirdNumber, sumOfNumbers; //code is being updated to print sum of 3 instead of 2 (edited by Daksh)
cout << "Enter three integers: ";
cin >> firstNumber >> secondNumber >>thirdNumber;
// sum of THREE numbers in stored in variable sumOfTwoNumbers
sumOfNumbers = firstNumber + secondNumber+thirdNumber;
// Prints sum
cout << firstNumber << " + " << secondNumber <<" + " << thirdNumber << " = " << sumOfNumbers<<endl;
  //I AM DEVANSH. I AM APPENDING A CODE THAT TAKES 4 ARRAY ELEMENTS AS INPUT AND THEN PRINTS THEIR SUM !!!!!!!!
  int arr[4];
  int sum=0;
  int product=1; // I am adding product also (edited by Daksh)
  cout<<"Enter 4 elements to sum up:"<<endl;
  for(int i=0;i<=3;i++)
  {
    cin>>arr[i];
    sum=sum+arr[i];
    product *= arr[i];

  }
  cout<<"Sum of the entered elements is: "<<sum<<endl;
  cout<<"Product of the entered elements is: "<<product<<endl;
return 0;
}
```

Checking the status:

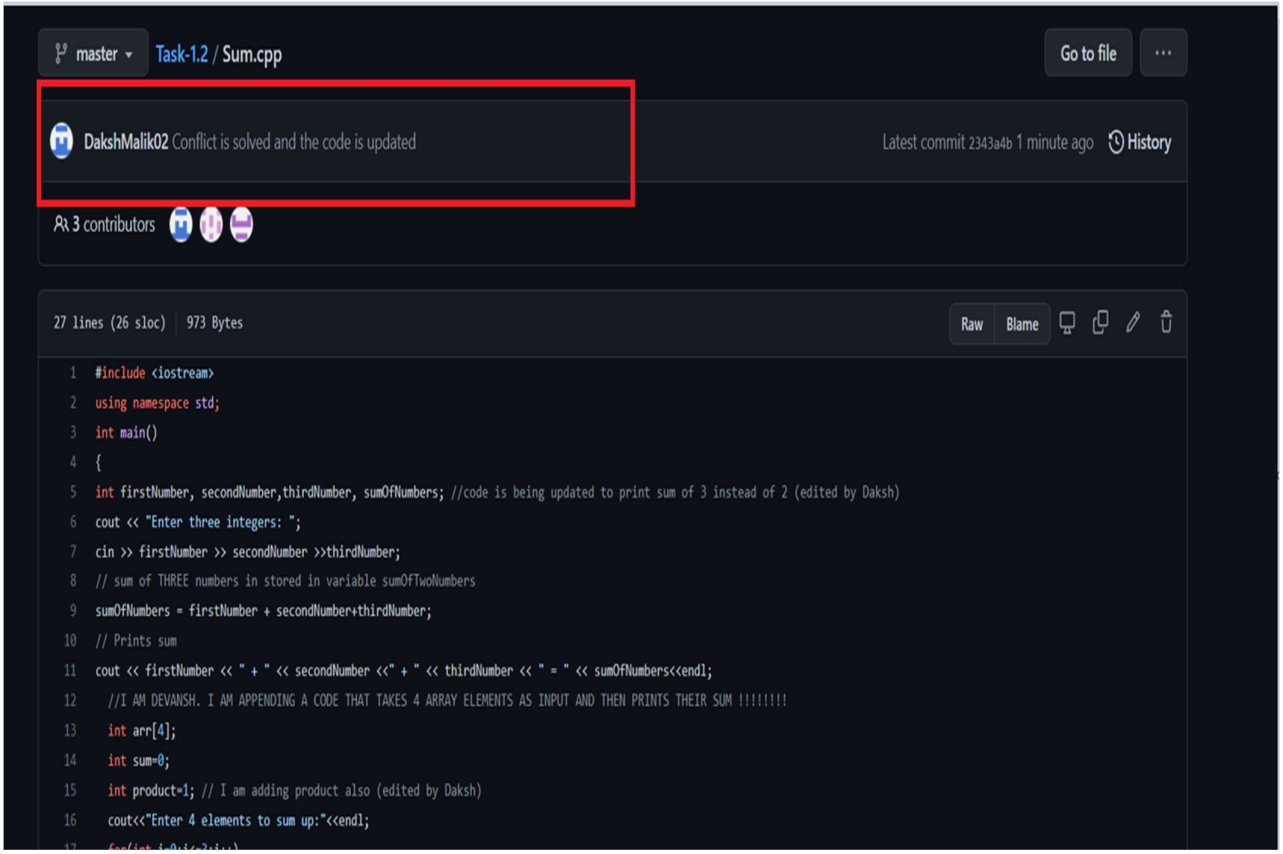
```
DAKSH MALIK@LAPTOP-BVBKAHE8 MINGW64 /d/Task 1.2/Task-1.2 (master)
$ git pull
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 1.03 KiB | 151.00 KiB/s, done.
From https://github.com/HarshKadiyan/Task-1.2
   d38656a..6707572  master     -> origin/master
Auto-merging Sum.cpp
CONFLICT (content): Merge conflict in Sum.cpp
Automatic merge failed; fix conflicts and then commit the result.

DAKSH MALIK@LAPTOP-BVBKAHE8 MINGW64 /d/Task 1.2/Task-1.2 (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:   Sum.cpp

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

```
DAKSH MALIK@LAPTOP-BVBKAHE8 MINGW64 /d/Task 1.2/Task-1.2 (master|MERGING)
$ git commit -m "Conflict is solved and the code is updated"
[master 2343a4b] Conflict is solved and the code is updated

DAKSH MALIK@LAPTOP-BVBKAHE8 MINGW64 /d/Task 1.2/Task-1.2 (master)
$ git push -u origin master
Enumerating objects: 10, done.
Counting objects: 100% (10/10), done.
Delta compression using up to 8 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (6/6), 818 bytes | 818.00 KiB/s, done.
Total 6 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 1 local object.
To https://github.com/HarshKadiyan/Task-1.2
   6707572..2343a4b  master -> master
branch 'master' set up to track 'origin/master'.
```

Finally checking the results on remote repository after pushing it:



Conflicts are resolved:

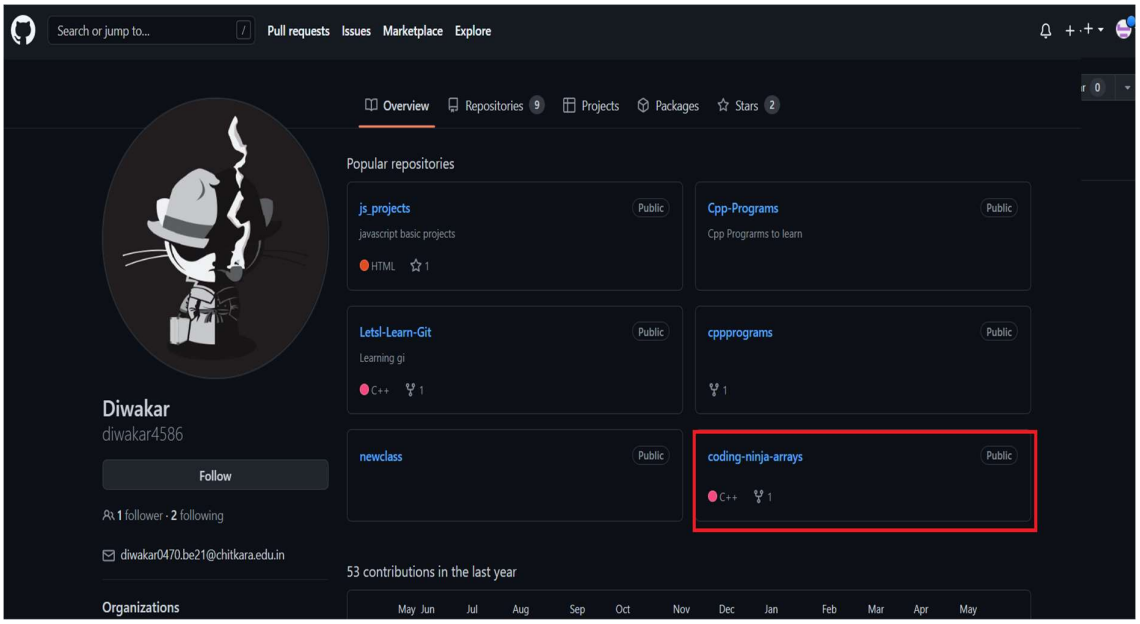


TASK 2:

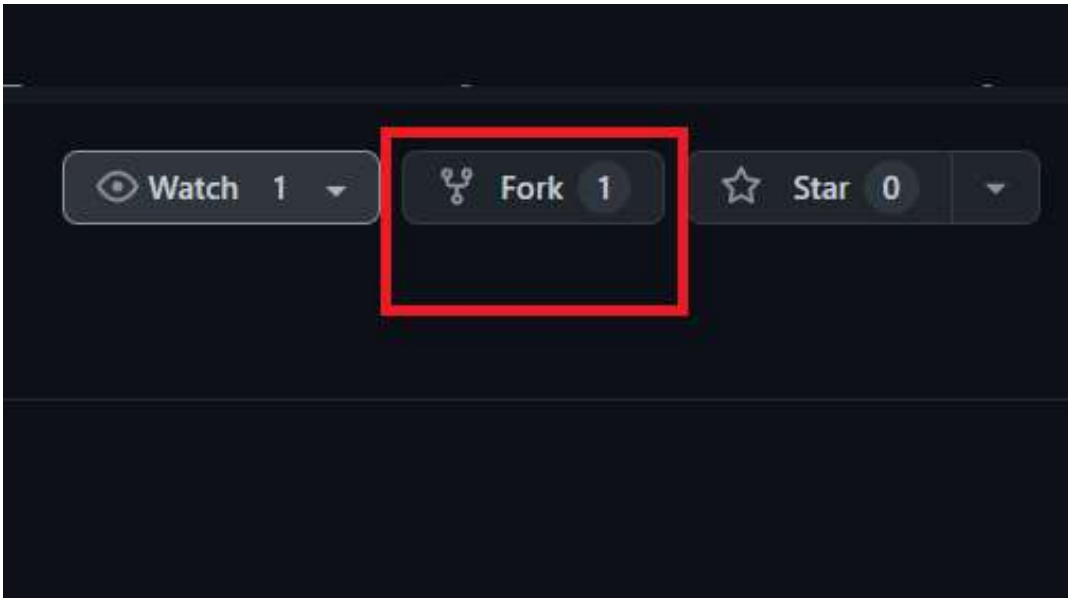
FORK AND COMMIT:

We have 3 team members other than me in the team so each team member will collaborate (fork and commit) in other team member's repo.

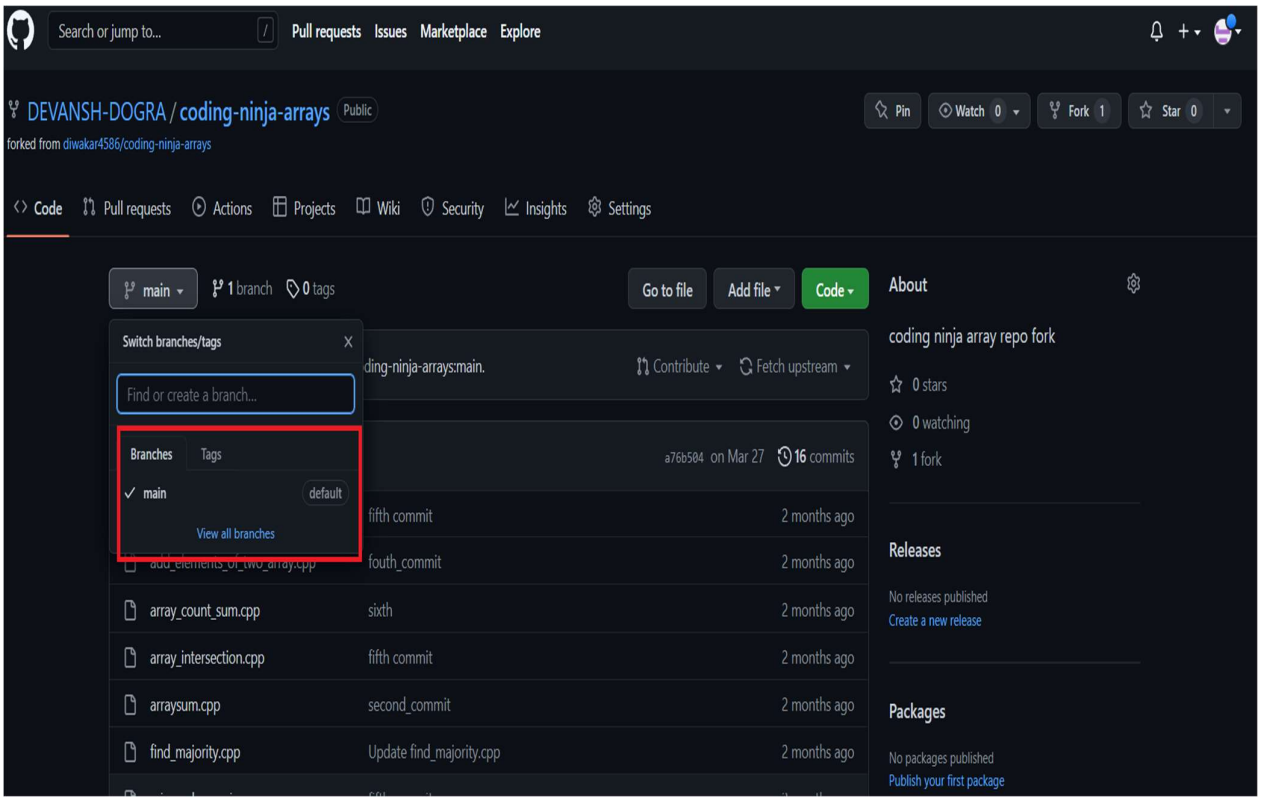
Forking is done on the GitHub Account while Cloning is done using Git. When you fork a repository, you create a copy of the original repository (upstream repository) but the repository remains on your GitHub account. Forking a repository allows you to freely test and debug with changes without affecting the original project. One of the excessive use of forking is to propose changes for bug fixing. Here we have started the group(collaborators) activity. Team member 1: As we can see in the below screenshot he has 9 repositories. I forked the repository" coding-ninja-arrays" to commit changes in it.



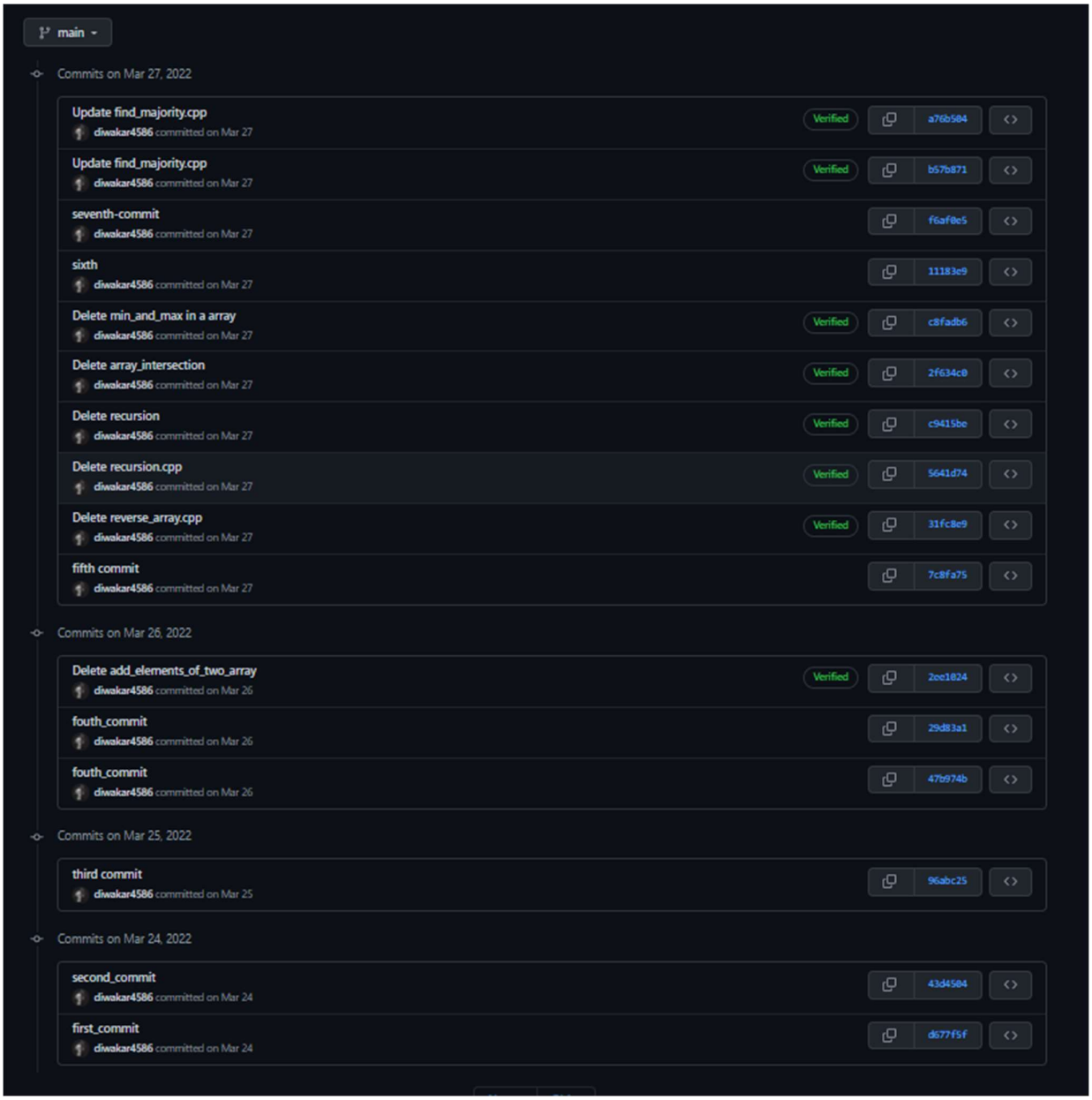
Forking is done by using the "Fork" button that is visible on opening the repository. After clicking on it, I successfully get a copy of this repository on my own git hub account. And this is now ready for us to clone and make our changes that we want to commit also.



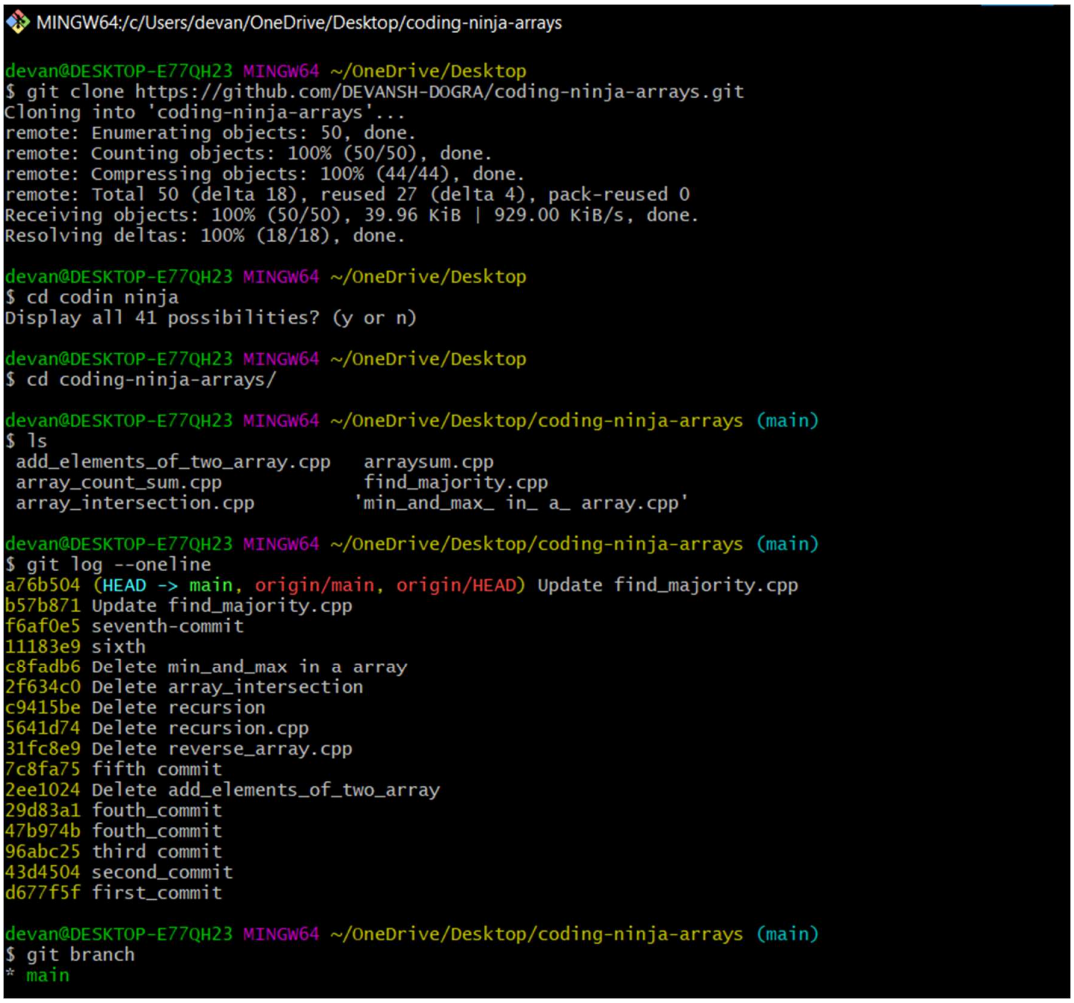
The below screenshot gives the information about the forked repository. The repository is now in my own git hub account DEVANSH-DOGRA. The forked repository has one fork, one branch that is the default one called the 'main' branch,it has 16 commits that have been done by my team member 1 and it has those files that can be seen in the picture.



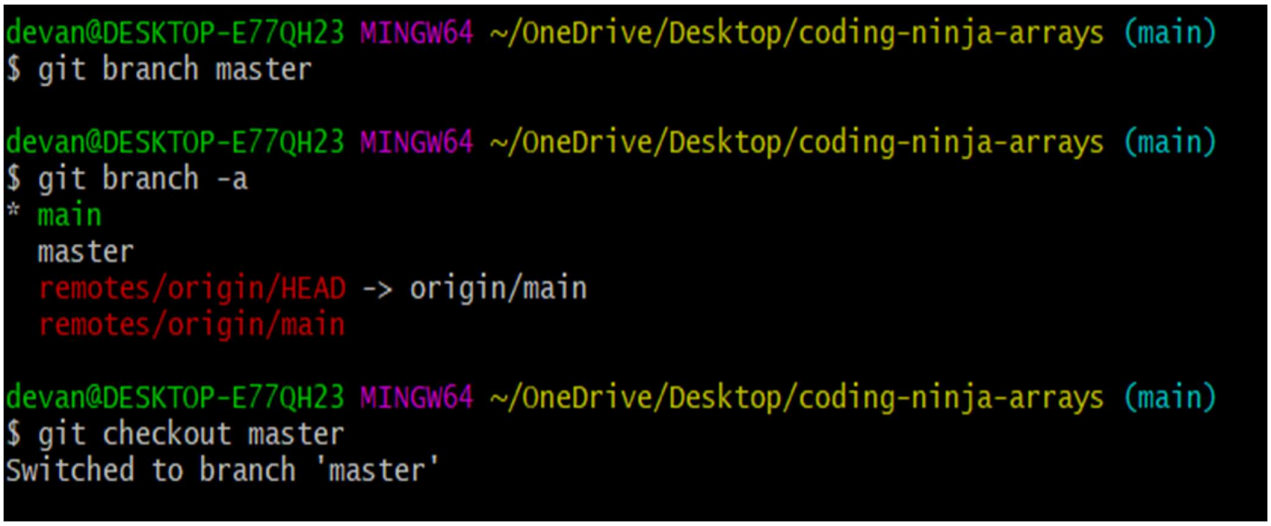
This is the list of the sixteen commits made by the team member 1:



Now, I am cloning the same repository into my own system(local) using the git clone command to edit it further and commit the changes.



A new branch is created named master in which I will make my desired changes.



Git log --oneline command displays those sixteen commits that were earlier made.

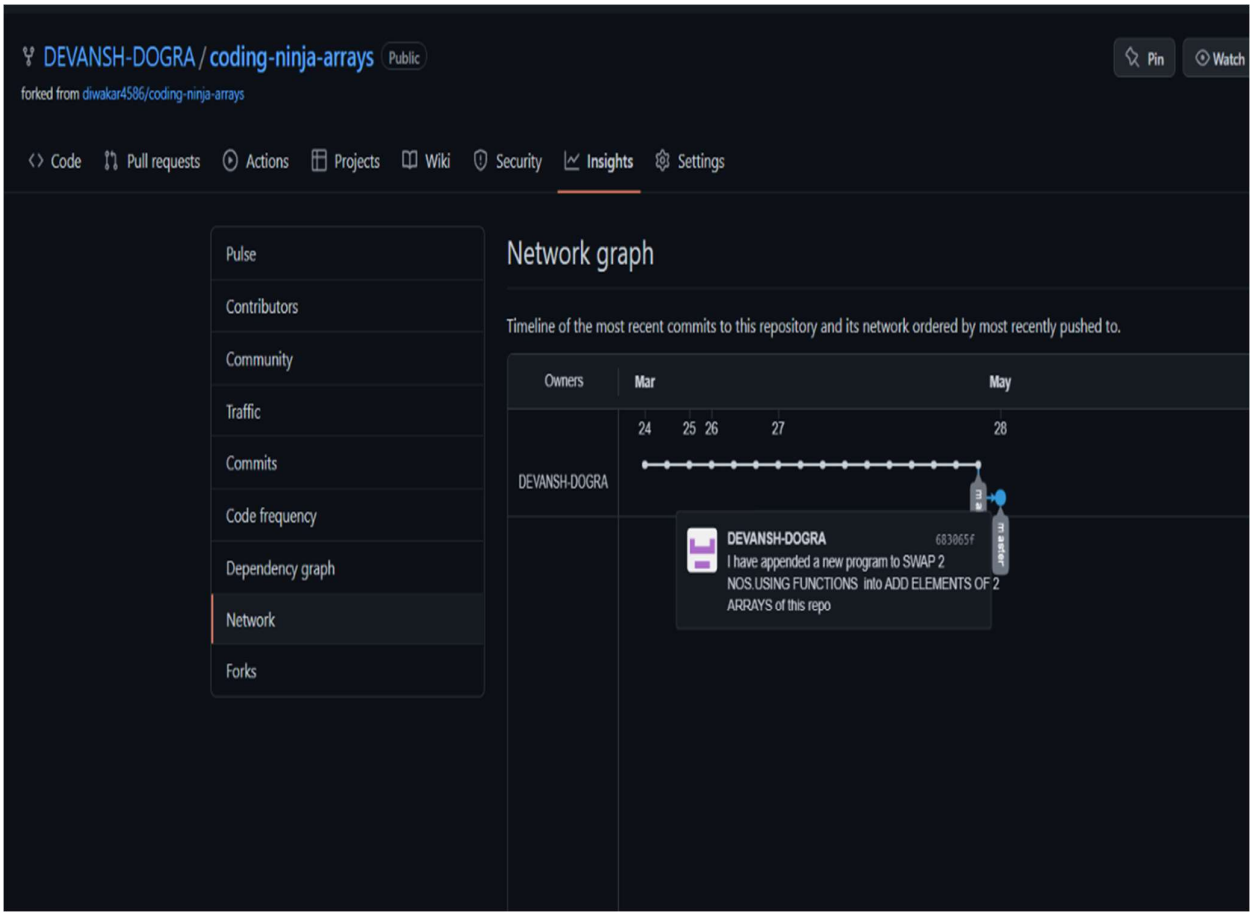
```
devan@DESKTOP-E77QH23 MINGW64 ~/OneDrive/Desktop/coding-ninja-arrays (master)
$ git log --oneline
a76b504 (HEAD -> master, origin/main, origin/HEAD, main) Update find_majority.cpp
b57b871 Update find_majority.cpp
f6af0e5 seventh-commit
11183e9 sixth
c8fadb6 Delete min_and_max in a array
2f634c0 Delete array_intersection
c9415be Delete recursion
5641d74 Delete recursion.cpp
31fc8e9 Delete reverse_array.cpp
7c8fa75 fifth commit
2ee1024 Delete add_elements_of_two_array
29d83a1 fouth_commit
47b974b fouth_commit
96abc25 third commit
43d4504 second_commit
d677f5f first_commit
```

Here the changes in the file “add_elements_of_two array.cpp” begin. The command vi filename is used to insert the new code. In this file I have appended a code that swaps two elements using a function.

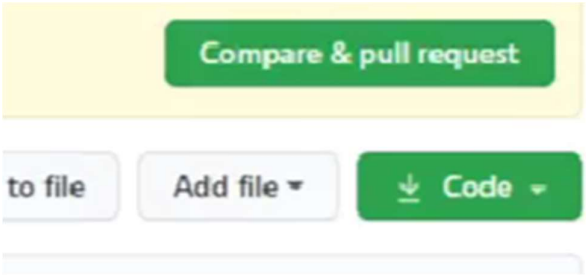
As the changes are committed. This file is going to be added to the staging area and then pushed to my own git hub account from where I can generate a pull request to team member 1 to merge into his account.

```
devan@DESKTOP-E77QH23 MINGW64 ~/OneDrive/Desktop/coding-ninja-arrays (master)
$ git add .

devan@DESKTOP-E77QH23 MINGW64 ~/OneDrive/Desktop/coding-ninja-arrays (master)
$ git commit -m "I have appended a new program to SWAP 2 NOS.USING FUNCTIONS  into ADD ELEMENTS OF 2 ARRAYS of this repo"
[master 683065f] I have appended a new program to SWAP 2 NOS.USING FUNCTIONS  into ADD ELEMENTS OF 2 ARRAYS of this repo
 2 files changed, 20 insertions(+), 5 deletions(-)
 create mode 100644 .add_elements_of_two_array.cpp.swp

devan@DESKTOP-E77QH23 MINGW64 ~/OneDrive/Desktop/coding-ninja-arrays (master)
$ git push -u origin master
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), 1.71 KiB | 584.00 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.
remote:
remote: Create a pull request for 'master' on GitHub by visiting:
remote:      https://github.com/DEVANSH-DOGRA/coding-ninja-arrays/pull/new/master
remote:
To https://github.com/DEVANSH-DOGRA/coding-ninja-arrays.git
 * [new branch]      master -> master
branch 'master' set up to track 'origin/master'.
```

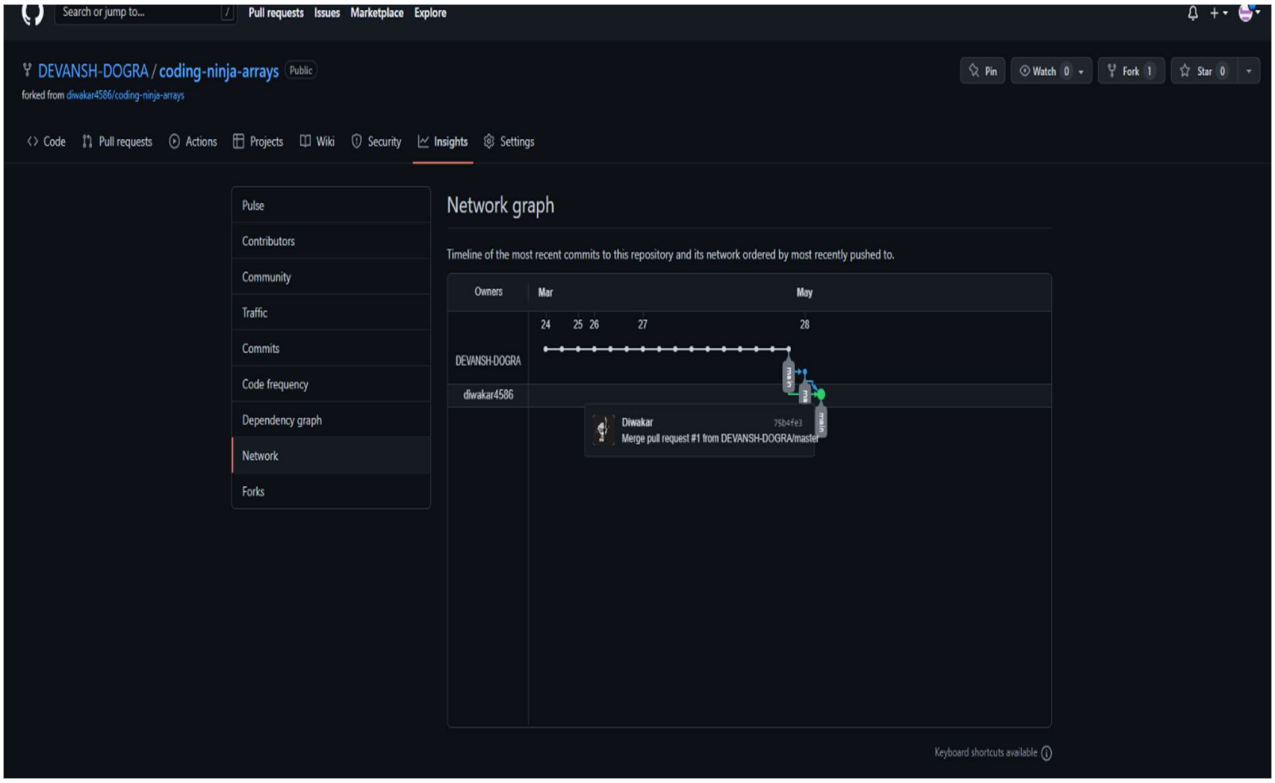
Below is the screen shot of the network graph that shows the new branch created and also commit and the commit message ”I have appended a new program to SWAP 2 NOS USING FUNCTION ”into “ADD ELEMENTS OF 2 ARRAYS” of this repository.



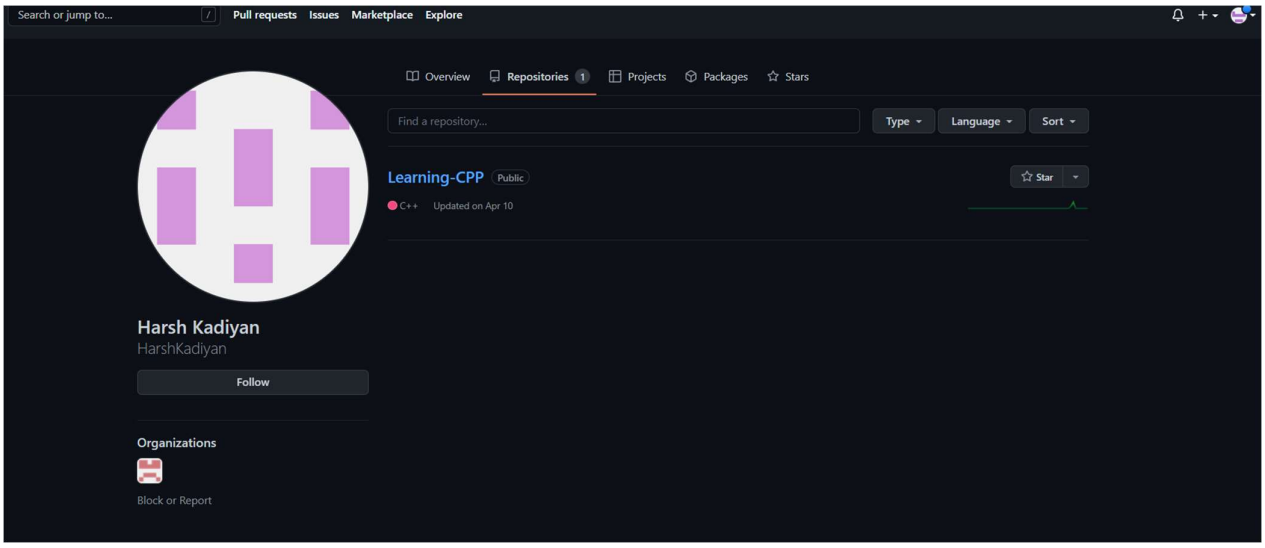
Now after pushing the repo on my git hub account we can see clearly that the Compare & pull request button takes a good shape! On clicking this button a pull request is generated that has to be merged by the team member on whose repo I have committed changes.



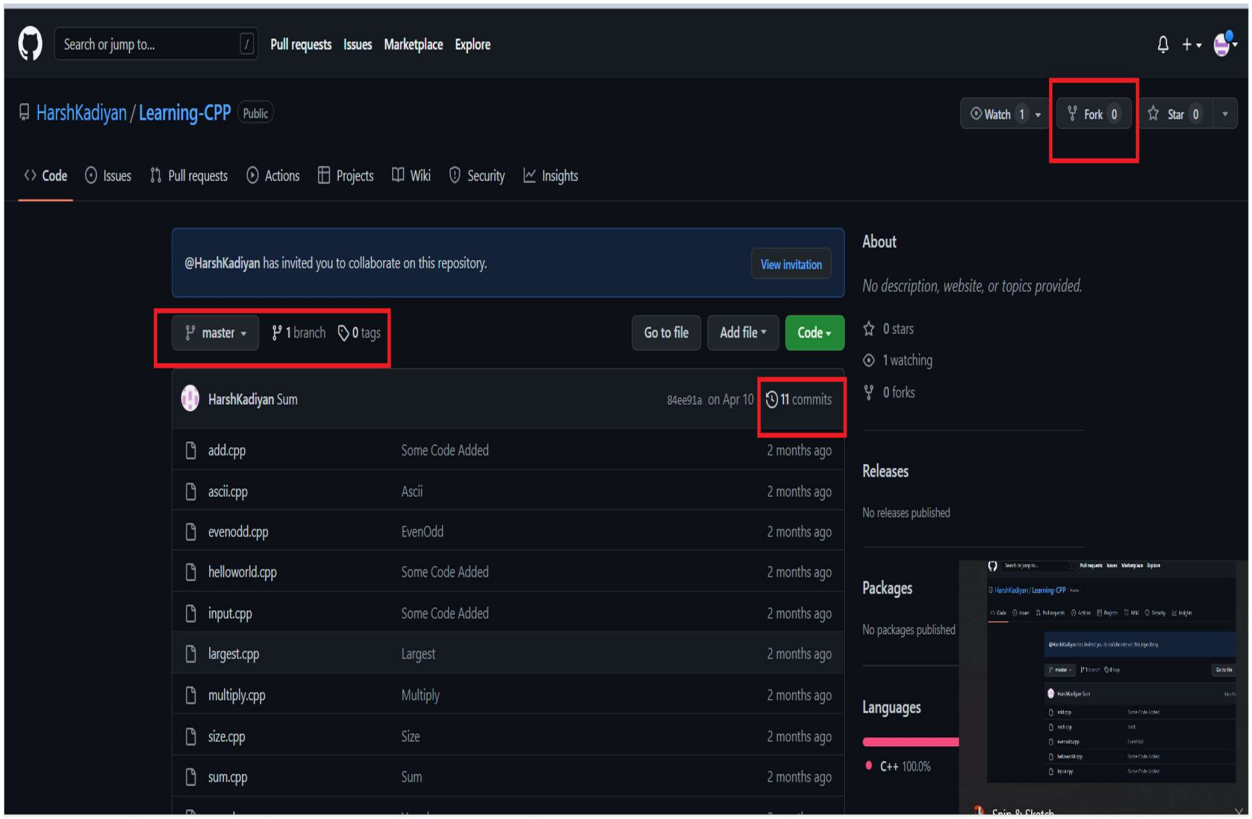
The below screenshot shows how the network graph of branches looks like when team member 1 merges the pull request I generated earlier.



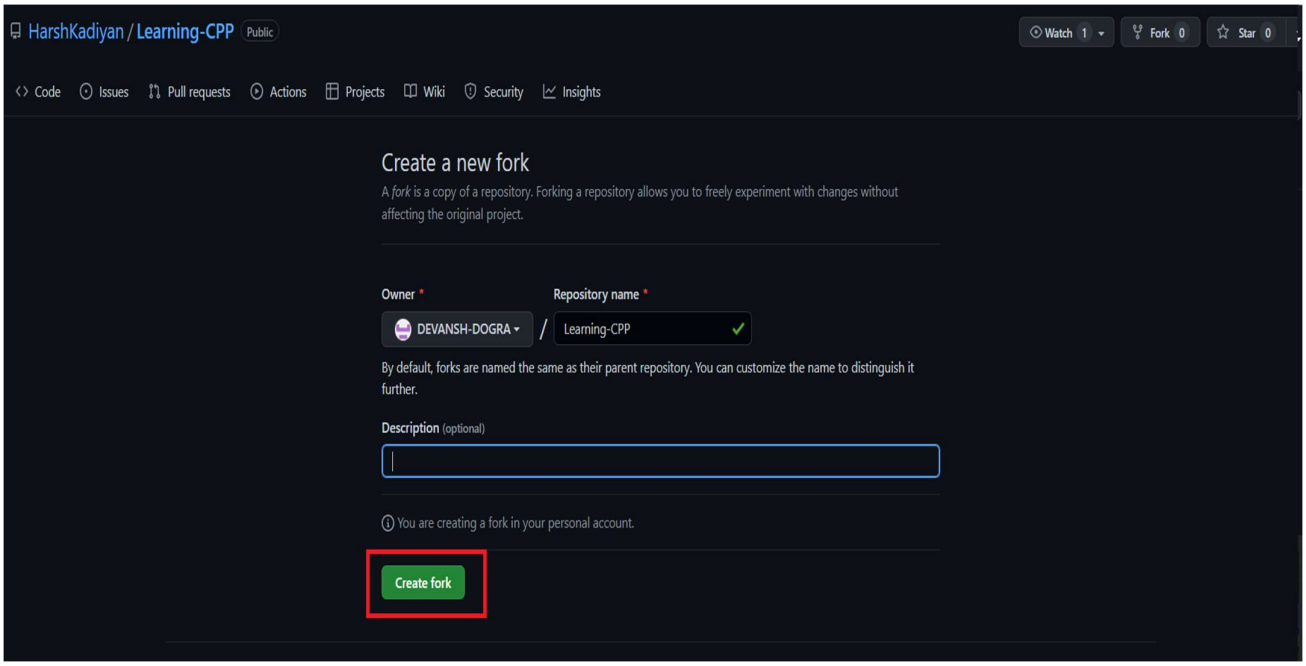
Forking and committing with team member 2: In the below screenshot we can clearly see that our team member 2 has only one repository made named Learning-CPP. So, I will be forking and committing my changes in this very repository only.

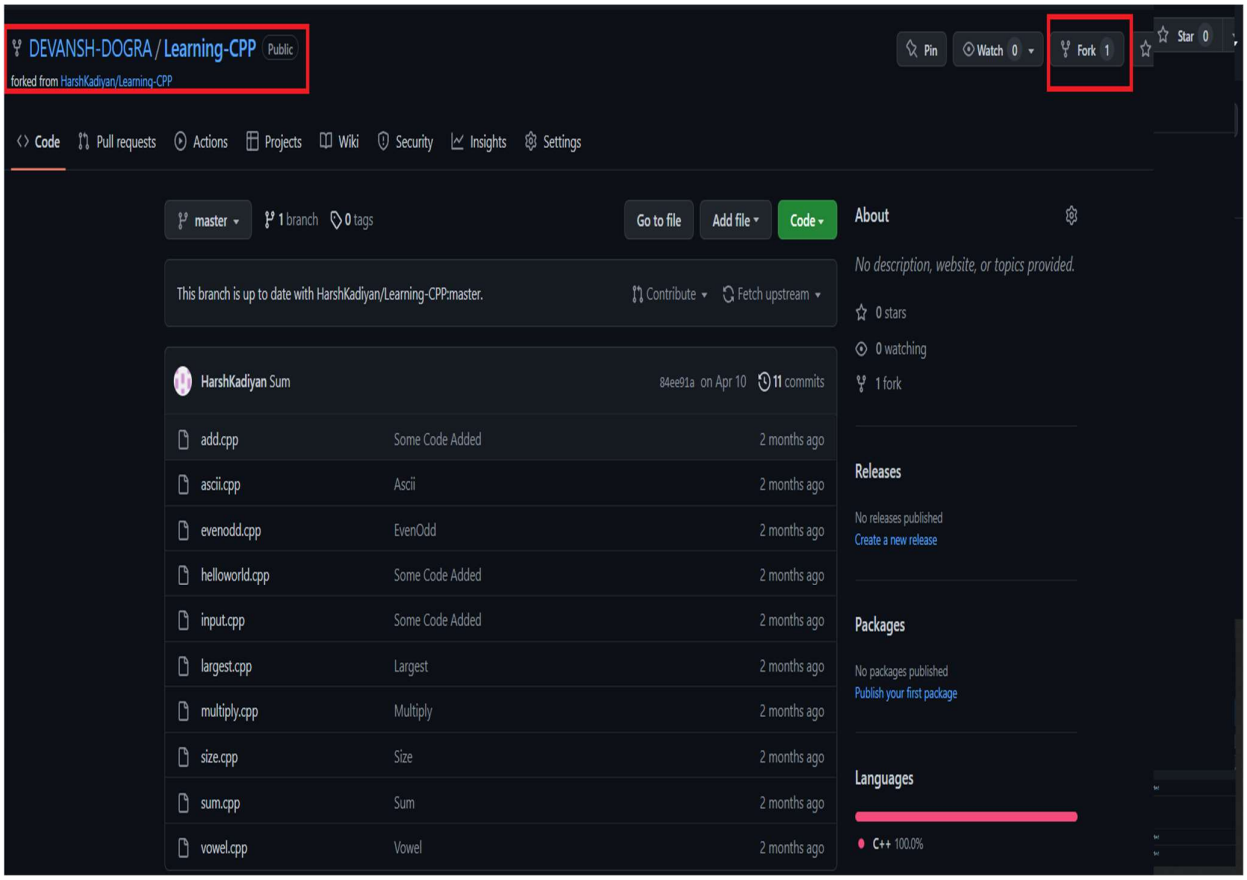


On opening this repository we can clearly see that it has only one branch i.e.the default branch, eleven commits and zero forks. By clicking the fork button I will fork this repository to my own git hub account.

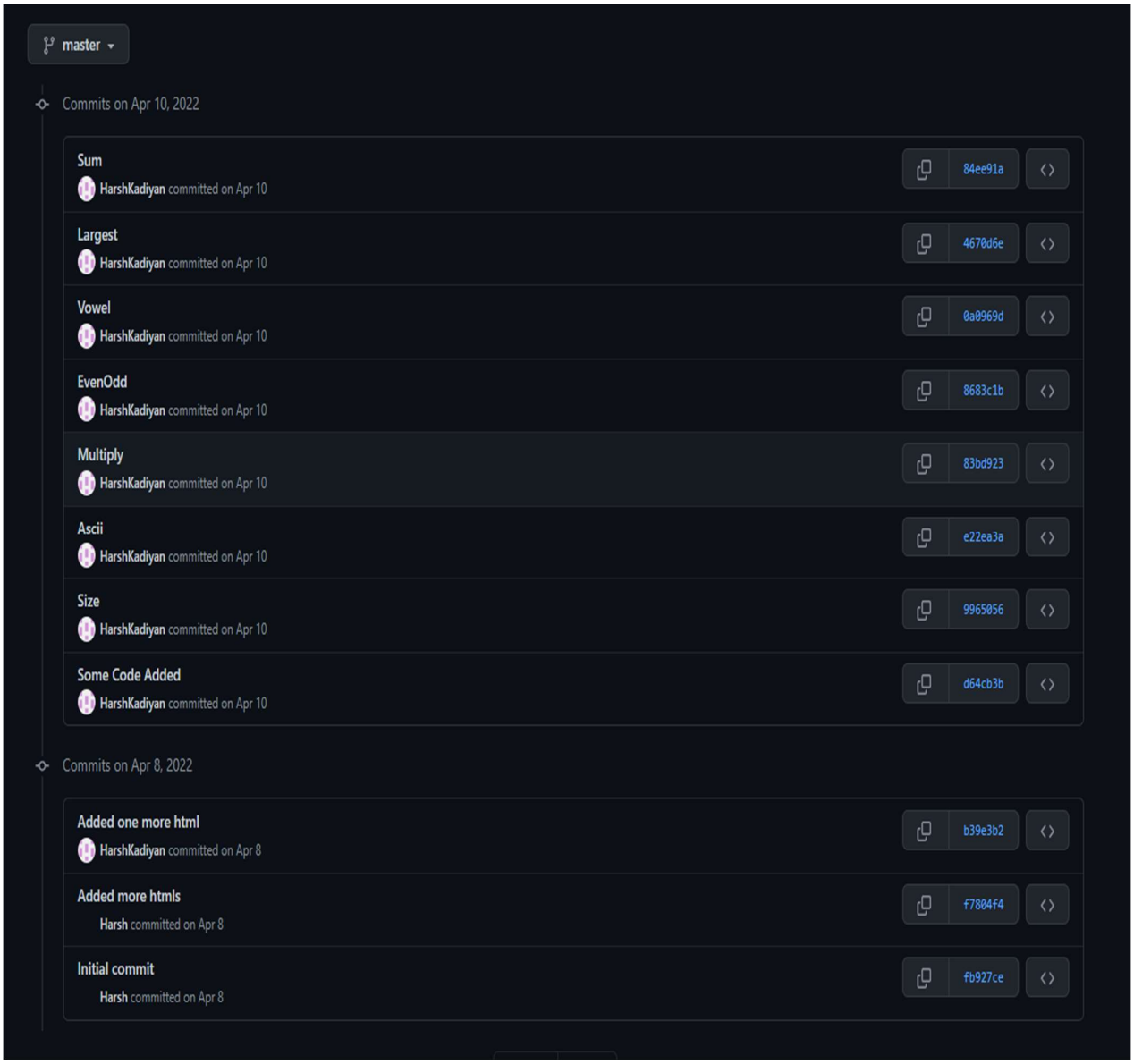


While forking we are asked to add a description message which is optional. By clicking the create fork button the repository will be forked into my git hub account.


HarshKadiyan / Learning-CPP
Public
Watch 1
Fork 0
Star 0
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Create a new fork
A fork is a copy of a repository. Forking a repository allows you to freely experiment with changes without affecting the original project.
Owner *
Repository name *
DEVANSH-DOGRA
Learning-CPP
By default, forks are named the same as their parent repository. You can customize the name to distinguish it further.
Description (optional)
You are creating a fork in your personal account.
Create fork


DEVANSH-DOGRA / Learning-CPP
Public
forked from HarshKadiyan/Learning-CPP
Pin
Watch 0
Fork 1
Star 0
Code
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
master
1 branch
0 tags
Go to file
Add file
Code
This branch is up to date with HarshKadiyan/Learning-CPP:master.
Contribute
Fetch upstream
HarshKadiyan Sum
84ee91a on Apr 10
11 commits
add.cpp Some Code Added 2 months ago
ascii.cpp Ascii 2 months ago
evenodd.cpp EvenOdd 2 months ago
helloworld.cpp Some Code Added 2 months ago
input.cpp Some Code Added 2 months ago
largest.cpp Largest 2 months ago
multiply.cpp Multiply 2 months ago
size.cpp Size 2 months ago
sum.cpp Sum 2 months ago
vowel.cpp Vowel 2 months ago
About
No description, website, or topics provided.
0 stars
0 watching
1 fork
Releases
No releases published
Create a new release
Packages
No packages published
Publish your first package
Languages
C++ 100.0%

The above screenshot clearly shows that the repository of team member 2 is successfully forked into my git hub account and now it has got one fork also. Below is the list of the eleven commits made by the second team member in the repository. This repository has only one branch named master.



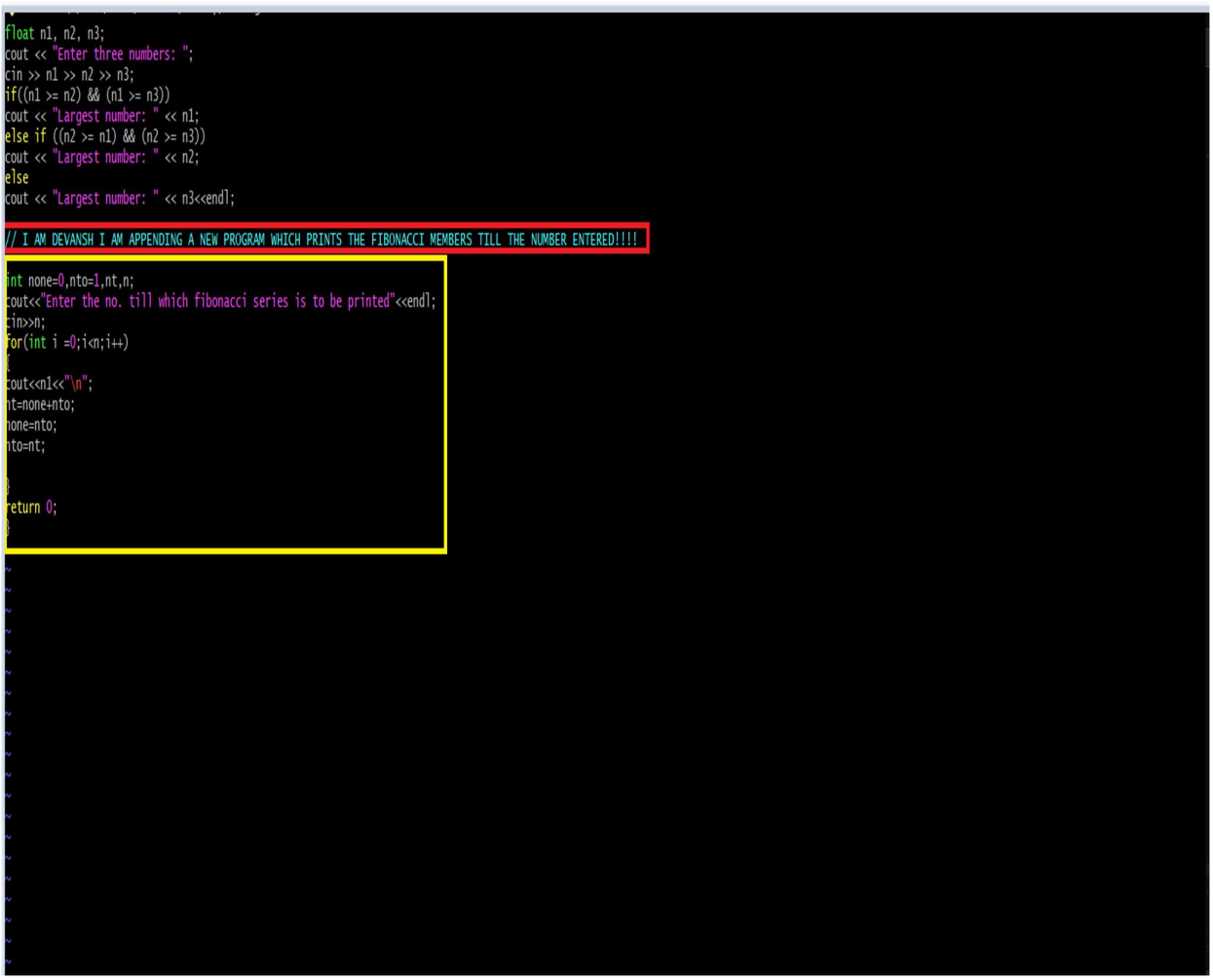
Now I will use git bash to clone this repository into my system(local). Git clone command is used to clone the repository. All the files in the repository is displayed using the ls command and all the commits are displayed using the git log --oneline. The branches that are currently in the repository are displayed using the git branch command.

All this is demonstrated in the below screenshot.

```
devan@DESKTOP-E77QH23 MINGW64 ~/OneDrive/Desktop
$ cd Learning-CPP/

devan@DESKTOP-E77QH23 MINGW64 ~/OneDrive/Desktop/Learning-CPP (master)
$ ls
add.cpp    evenodd.cpp     input.cpp    multiply.cpp  sum.cpp
ascii.cpp  helloworld.cpp  largest.cpp  size.cpp      vowel.cpp

devan@DESKTOP-E77QH23 MINGW64 ~/OneDrive/Desktop/Learning-CPP (master)
$ git log --oneline
84ee91a (HEAD -> master, origin/master, origin/HEAD) Sum
4670d6e Largest
0a0969d Vowel
8683c1b EvenOdd
83bd923 Multiply
e22ea3a Ascii
9965056 Size
d64cb3b Some Code Added
b39e3b2 Added one more html
f7804f4 Added more htmls
fb927ce Initial commit

devan@DESKTOP-E77QH23 MINGW64 ~/OneDrive/Desktop/Learning-CPP (master)
$ git branch
* master
```

Now, a new branch named feature is created in which I will make my desired changes. I will checkout to the new branch i.e., the feature branch using the git checkout command.The command git branch -a displays all the branches that have been created in this repository.

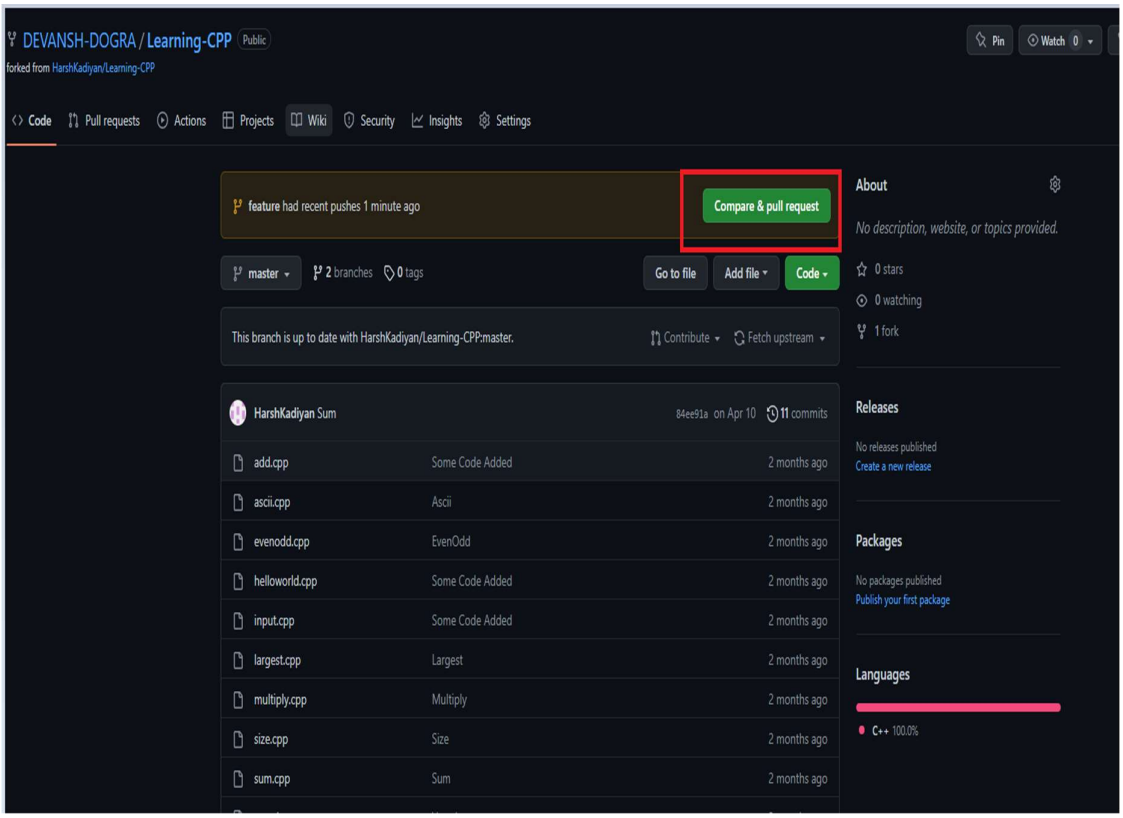
```
devan@DESKTOP-E77QH23 MINGW64 ~/OneDrive/Desktop/Learning-CPP (master)
$ git branch
* master

devan@DESKTOP-E77QH23 MINGW64 ~/OneDrive/Desktop/Learning-CPP (master)
$ git branch feature

devan@DESKTOP-E77QH23 MINGW64 ~/OneDrive/Desktop/Learning-CPP (master)
$ git checkout feature
Switched to branch 'feature'

devan@DESKTOP-E77QH23 MINGW64 ~/OneDrive/Desktop/Learning-CPP (feature)
$ git branch -a
* feature
  master
  remotes/origin/HEAD -> origin/master
  remotes/origin/master
```



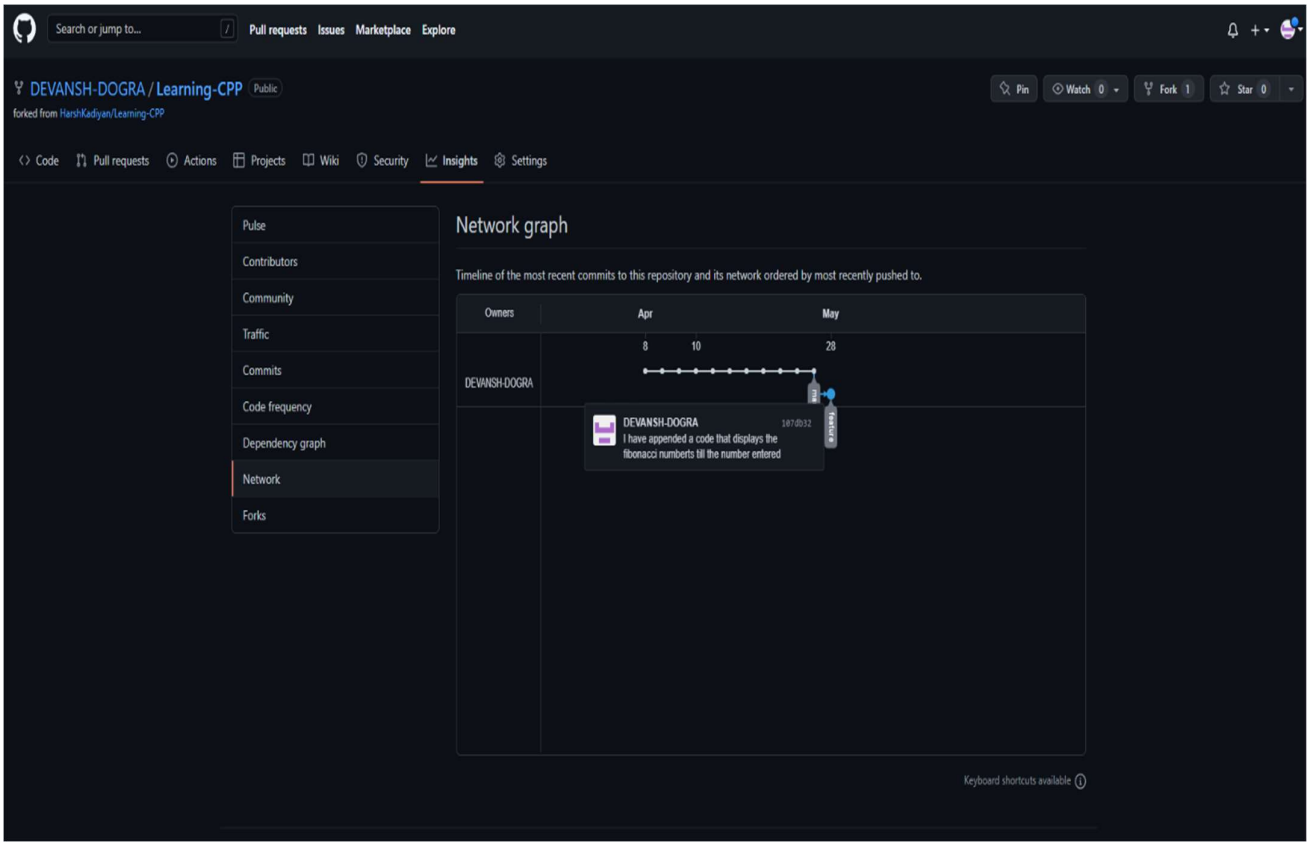
Now, as I have completed my desired changes, I will commit those changes with a reference commit message and then push it into my remote repository.

```
devan@DESKTOP-E77QH23 MINGW64 ~/OneDrive/Desktop/Learning-CPP (feature)
$ git add .

devan@DESKTOP-E77QH23 MINGW64 ~/OneDrive/Desktop/Learning-CPP (feature)
$ git commit -m"I have appended a code that displays the fibonacci numberts till the number entered"
[feature 107db32] I have appended a code that displays the fibonacci numberts till the number entered
 3 files changed, 17 insertions(+), 2 deletions(-)
 create mode 100644 largest.exe
 create mode 100644 largest.o

devan@DESKTOP-E77QH23 MINGW64 ~/OneDrive/Desktop/Learning-CPP (feature)
$ git push -u origin feature
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 8 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 41.85 KiB | 3.22 MiB/s, done.
Total 5 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
remote:
remote: Create a pull request for 'feature' on GitHub by visiting:
remote:      https://github.com/DEVANSH-DOGRA/Learning-CPP/pull/new/feature
remote:
To https://github.com/DEVANSH-DOGRA/Learning-CPP.git
 * [new branch]      feature -> feature
branch 'feature' set up to track 'origin/feature'.
```

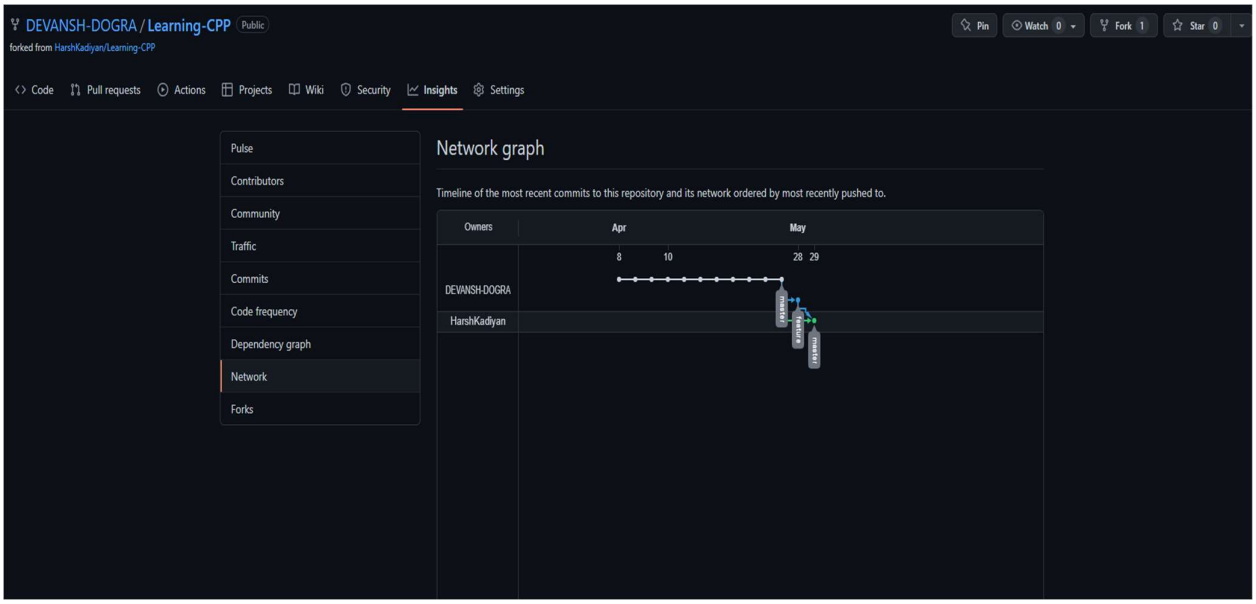
After I pushed into the remote repository, a compare & pull request button takes a good shape as shown in the below picture.On clicking this button, a pull request is successfully generated.



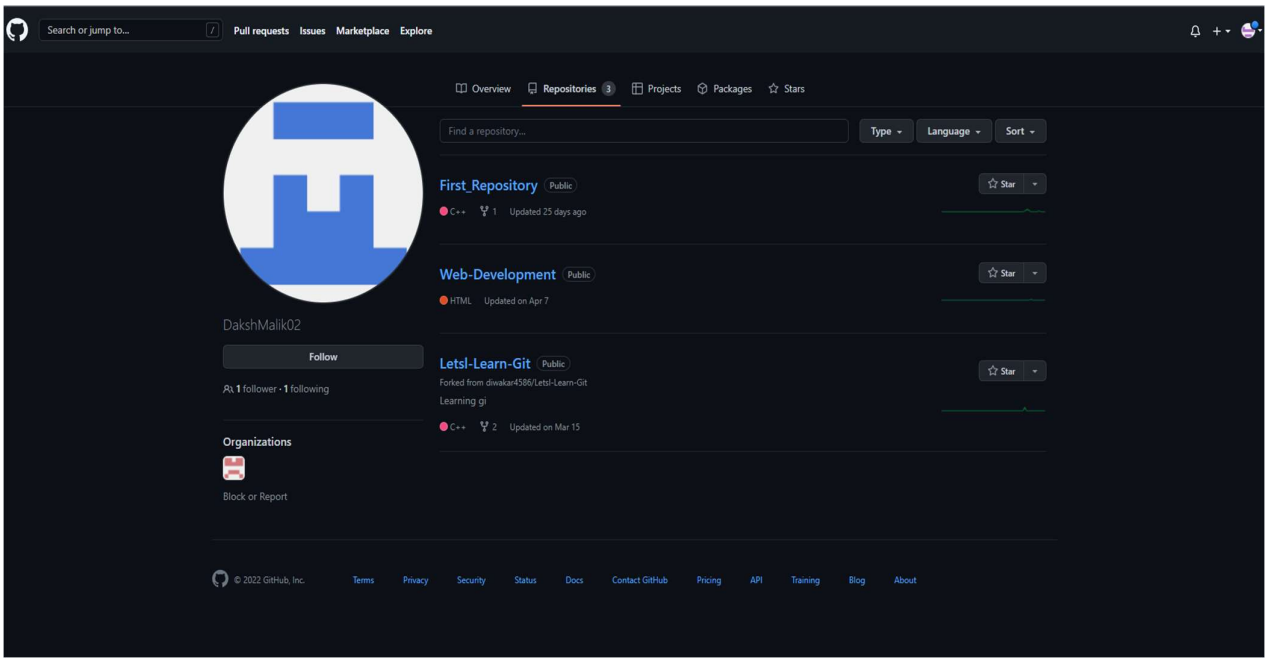
The pull request has been generated successfully to the team member 2.Below is the screenshot of the network graph that shows the new branch created and also commit and the commit message ”I have appended a code that displays the Fibonacci numbers till the number entered ” of this repository.



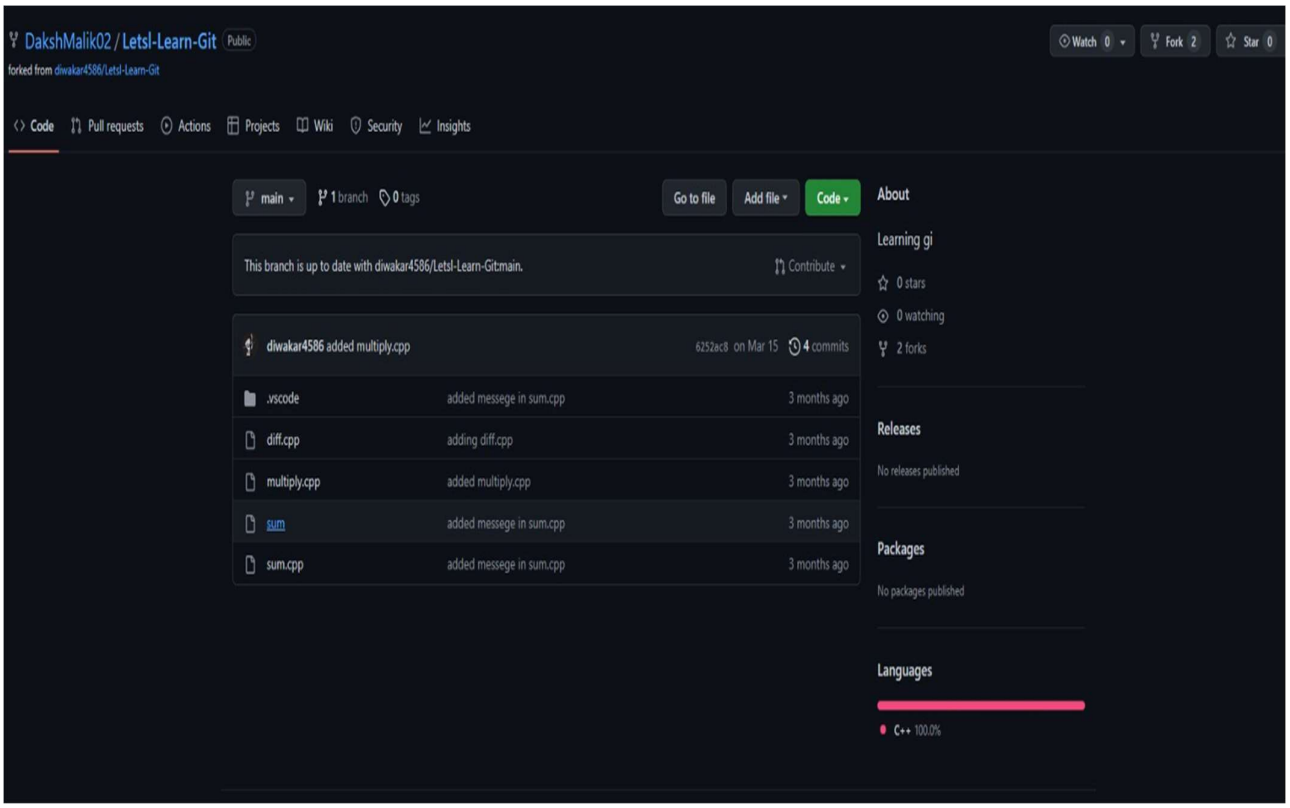
The below screenshot shows how the network graph of branches looks like when team member 2 merges the pull request I generated earlier.



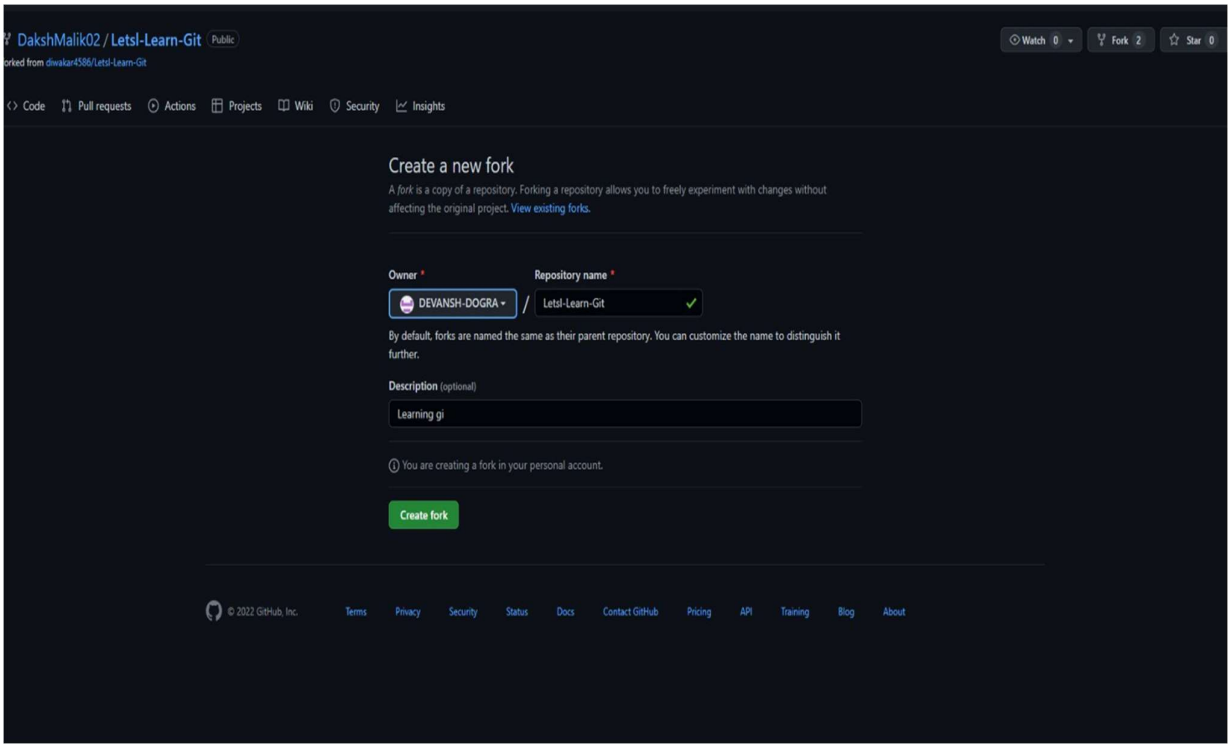
Forking and committing with team member 3: In the below screenshot we can clearly see that our team member 3 has three repositories made. So, I will fork and commit my changes in “Letsl-Learn-Git”.



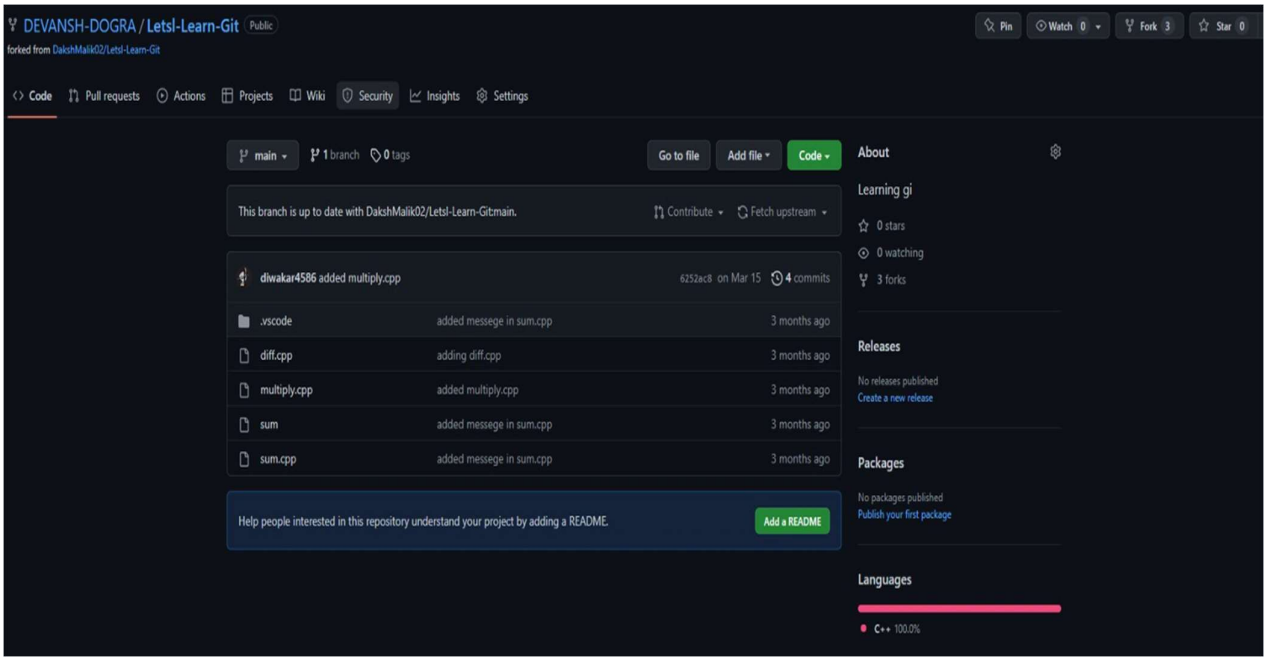
On opening this repository (Letsl-Learn-Git) we can clearly see that it has one branch named main(default branch) four commits made by the team member 3 and two forks.



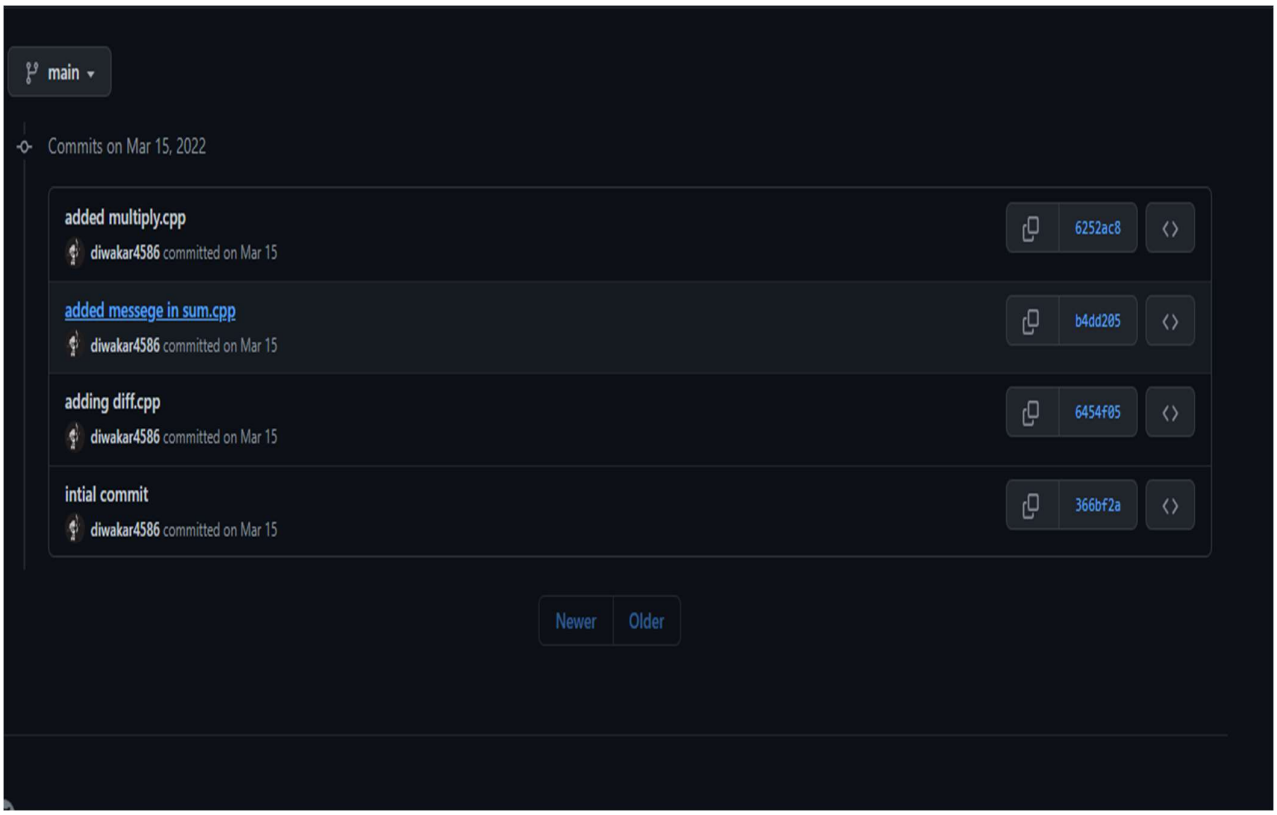
Now I will fork this repository into my own git hub account using the fork button.



After clicking on the fork button it asks to add a description message but its optional and repository will be successfully forked just by doing the last step i.e., clicking on “Create fork” button.



The above screenshot clearly shows that “Letsl-Learn-Git” is successfully cloned int my git hub account and now it has got three forks.Below is the list of the four commits made by the team member 3 in his repository on the main branch :

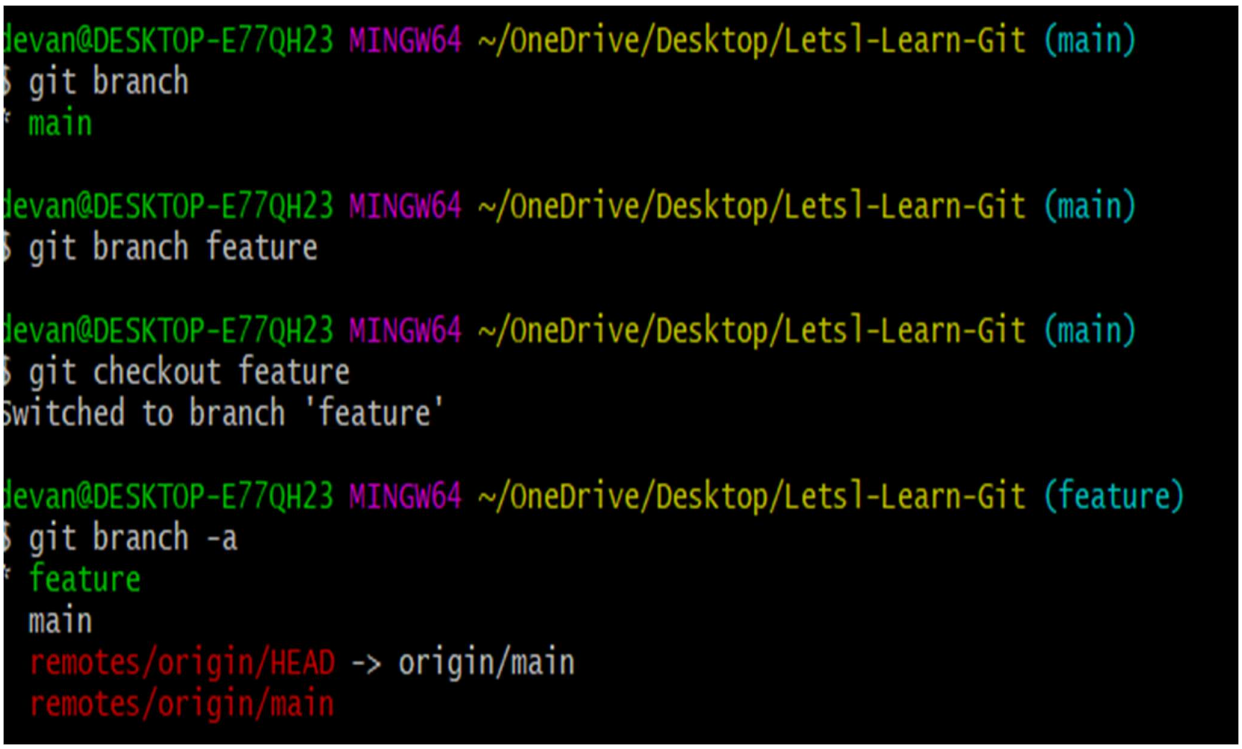


Now I will use git bash to clone this repository into my system(local). Git clone command is used to clone the repository. All the files in the repository is displayed using the ls command and all the commits are displayed using the git log --oneline. The branches that are currently in the repository are displayed using the git branch command.

```
devan@DESKTOP-E77QH23 MINGW64 ~/OneDrive/Desktop
$ git clone https://github.com/DEVANSH-DOGRA/Letsl-Learn-Git.git
Cloning into 'Letsl-Learn-Git'...
remote: Enumerating objects: 16, done.
remote: Counting objects: 100% (16/16), done.
remote: Compressing objects: 100% (13/13), done.
remote: Total 16 (delta 2), reused 16 (delta 2), pack-reused 0
Receiving objects: 100% (16/16), 15.21 KiB | 741.00 KiB/s, done.
Resolving deltas: 100% (2/2), done.
```

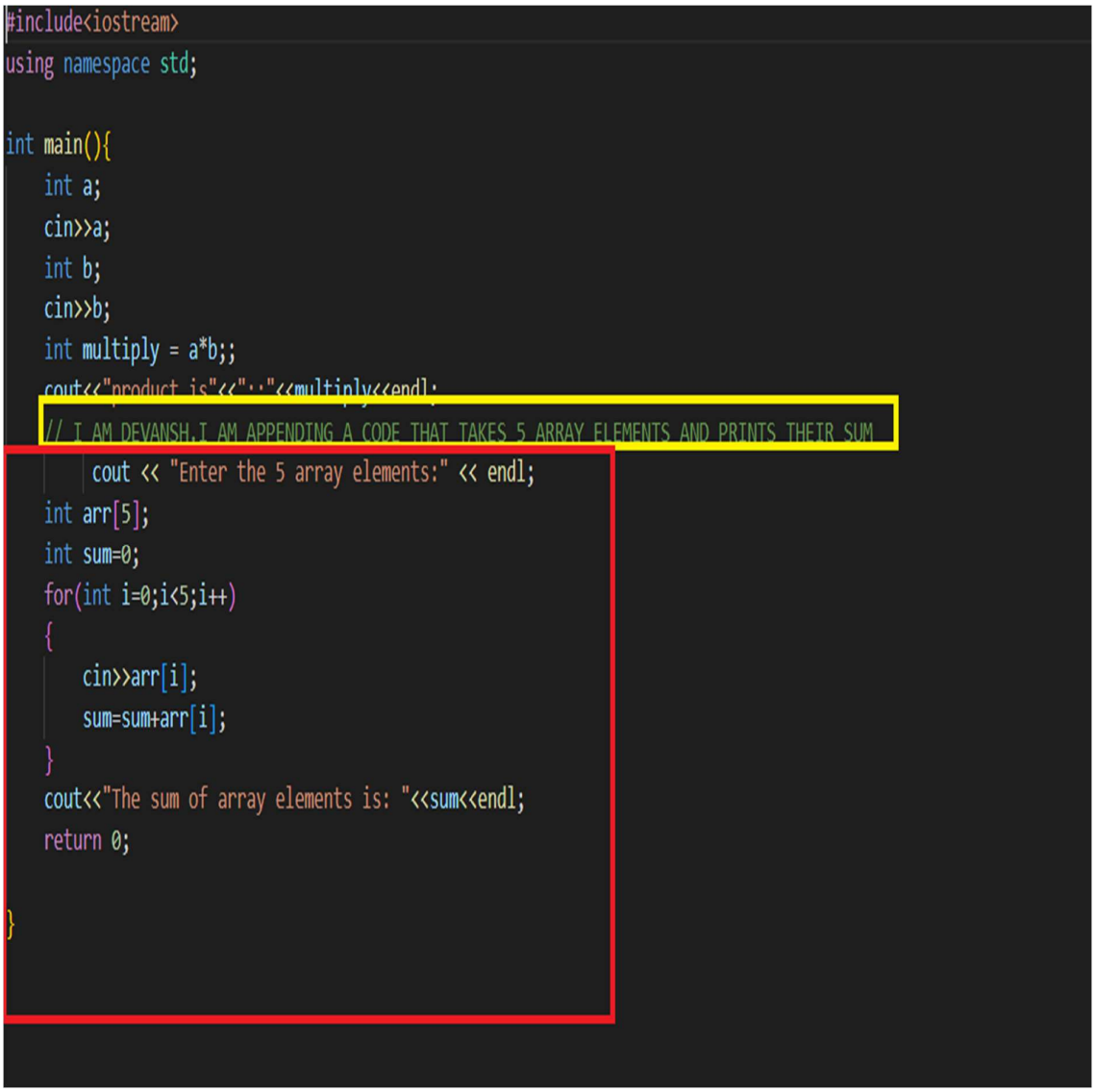
Now, a new branch named feature is created in which I will make my desired changes. I will checkout to the new branch i.e., the feature branch using the git checkout command.

```
devan@DESKTOP-E77QH23 MINGW64 ~/OneDrive/Desktop
$ cd Letsl-Learn-Git/

devan@DESKTOP-E77QH23 MINGW64 ~/OneDrive/Desktop/Letsl-Learn-Git (main)
$ ls
diff.cpp  multiply.cpp  sum  sum.cpp

devan@DESKTOP-E77QH23 MINGW64 ~/OneDrive/Desktop/Letsl-Learn-Git (main)
$ git log --oneline
6252ac8 (HEAD -> main, origin/main, origin/HEAD) added multiply.cpp
b4dd205 added messege in sum.cpp
6454f05 adding diff.cpp
366bf2a intial commit

devan@DESKTOP-E77QH23 MINGW64 ~/OneDrive/Desktop/Letsl-Learn-Git (main)
$ git branch
* main
```



Now, I will make my desired changes in the “multiply” file. The changes in the code file are done locally on compiler(VS Code).

Below is the screenshot of the changes done by me.

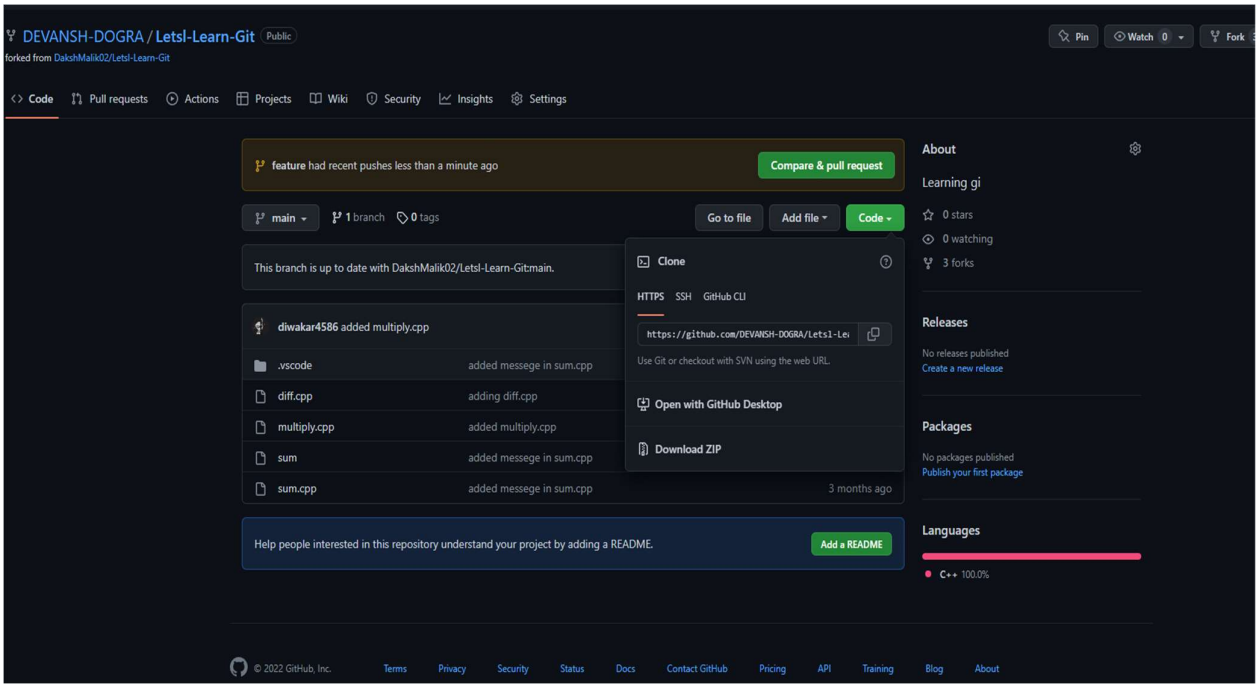


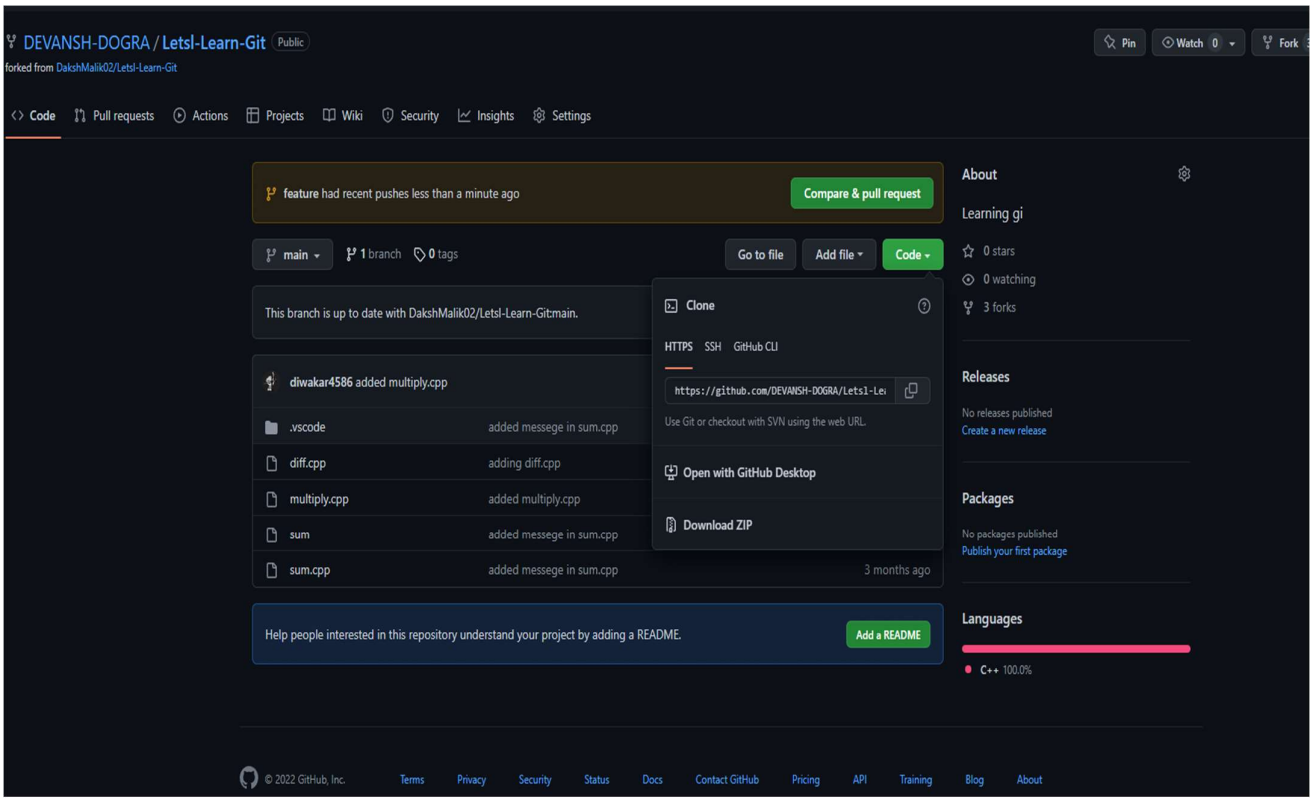
Now, as I have completed my desired changes, I will commit those changes with a reference commit message and then push it into my remote repository.

```
devan@DESKTOP-E77QH23 MINGW64 ~/OneDrive/Desktop/Letsl-Learn-Git (feature)
$ git add .

devan@DESKTOP-E77QH23 MINGW64 ~/OneDrive/Desktop/Letsl-Learn-Git (feature)
$ git commit -m"appended a code that takes 5 array elements and prints their sum
"
[feature 07cc1c2] appended a code that takes 5 array elements and prints their s
um
 3 files changed, 13 insertions(+), 3 deletions(-)
 create mode 100644 multiply.exe
 create mode 100644 multiply.o

devan@DESKTOP-E77QH23 MINGW64 ~/OneDrive/Desktop/Letsl-Learn-Git (feature)
$ git push -u origin feature
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 8 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 41.62 KiB | 3.78 MiB/s, done.
Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'feature' on GitHub by visiting:
remote:      https://github.com/DEVANSH-DOGRA/Letsl-Learn-Git/pull/new/feature
remote:
To https://github.com/DEVANSH-DOGRA/Letsl-Learn-Git.git
 * [new branch]      feature -> feature
branch 'feature' set up to track 'origin/feature'.
```

After I pushed into the remote repository, a compare & pull request button takes a good shape as shown in the below picture.

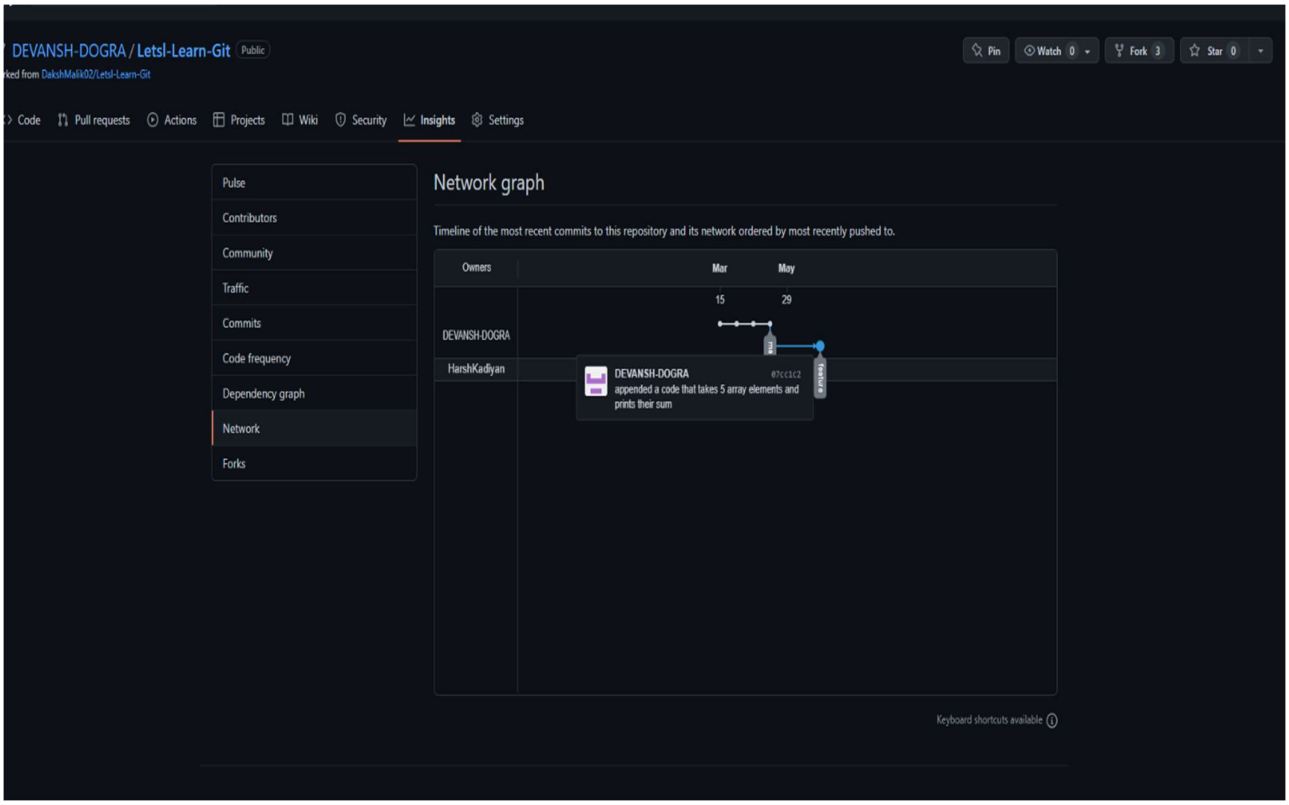
On clicking this button, a pull request is successfully generated.



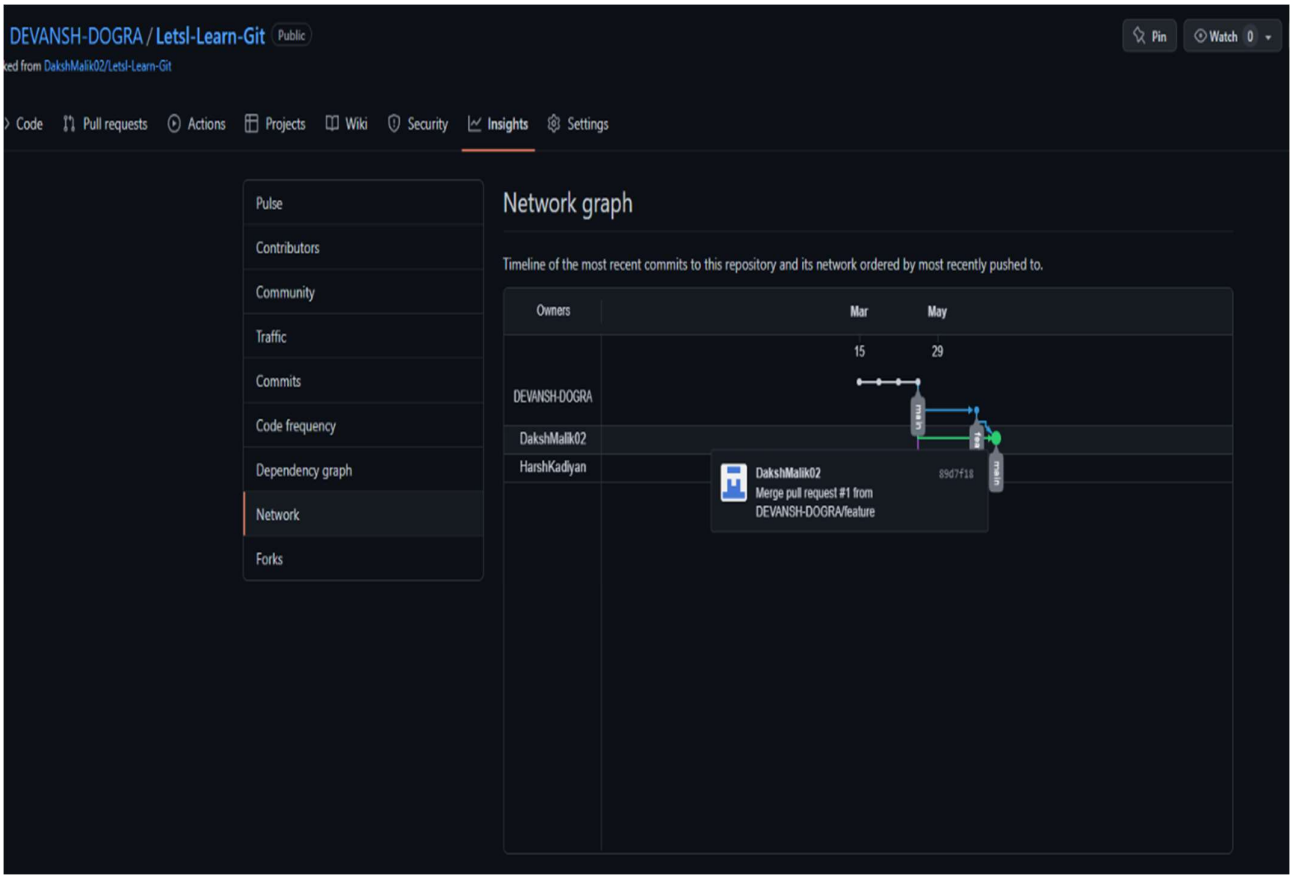


The pull request has been generated successfully to the team member 3.

Below is the screenshot of the network graph that shows the new branch created and also commit and the commit message.

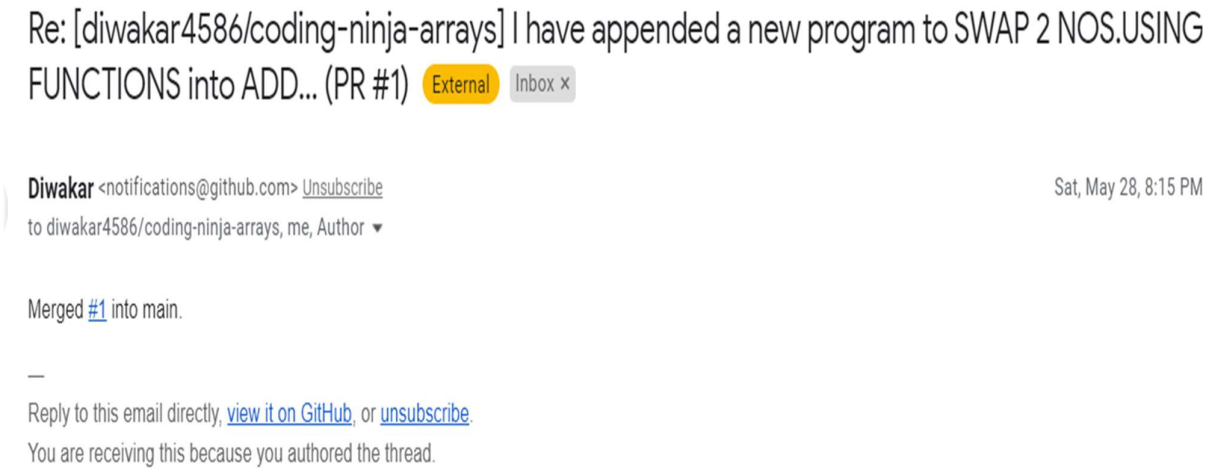


The below screenshot shows how the network graph of branches looks like when team member 3 merges the pull request I generated earlier.



The following are the types of alerts on the registered email id that we get when someone merges our pull requests:

When team member 1 merges my pull request into main branch of his repository:



When team member 2 merges my pull request into the master branch of his repository:

When team member 3 merges my pull request into the main branch of his repository: