SCM FILE

Source Code Management

SUBMITTED TO:

MONIT SIR

Submitted By: Aayush Mittal

Roll No: 2110990030

Cluster: Beta

Index

S. No	Experiment Name	Page No.
1.	Introduction	3-7
2	Setting up Github Client	8-13
3.	Setting Up Github Account	14-19
4.	Generate Logs	20
5.	Git Branching	21-23
6.	Git Life Cycle Description	24-27
7.	Add collaborators On Github Repository	28-33

8	Fork and Commit	34-42
9	Merge and Resolve Conflict	42-51
10	Reset and Revert	52-60
11	Create a Distributed Repository and Add Members in a Project Team	61-68
12	Open and Close Pull Request	69-78
13	Create a Pull Request on a Team Members Repository and close Pull Requests generated by Team Members on your own Repository as a Maintainer	79-93
14	Publish and Print Network Graphs	94-96

INTRODUCTION

Git

Git is a free and open source distributed version control system that records changes to a file or a set of files overtime so that you can recall specific versions later. The changes are stored in special database called repository. It was created by Linus Torvalds in 2005. It provides the flexibility to view source code according to user's need.

GitHub

GitHub is an online portal or a cloud-based online service that allows users to keep a track of the files. It enables developers to upload their

own code files and to collaborate with fellow developers on open-source projects.

GitHub also serves as a social networking site in which developers can openly network, collaborate, and pitch their work.

GIT VS GITHUB

Git	GitHub
1. Installed Locally.	1. Hosted in Cloud.
2. First Release in 2005.	2.Company Launched in 2008.
3. Maintained by Linux Foundation.	3. Purchased by Microsoft in 2018.
4. Focused on version control and code sharing.	4. Focused on centralised source code hosting.
5. Primarily a command-line Tool.	5. Administered through the web.
6. Provides Desktop Interface named git GUI.	6. Desktop Interface named GitHub Desktop.
7. No user Management Features.	7. Built in User management.
8. Minimal external tool configuration features.	8. Active Marketplace for tool Integration.
9. Competes with mercurial, subversion IBM	9. Competes with Atlassian bit bucket and Gitlab.
10. Open Source Licence.	10. Include a free tier and pay for use tier.

www.javatpoint.com/git-vs-github

What is Repository?

In git, the repository is like a data structure used by VCS to store metadata for a set of files and directories. It contains a collection of files as well as the history of changes made to those files. Repository in Git is considered as your project folder and repository has all project related data

What is Version Control System (VCS)

- **Centralized Version Control System**

In a centralized version control system (CVCS), server acts as the main repository which stores every version of code. Using centralized source control, every user commits directly to the main branch, so this type of version control often works well for small teams, because team members have the ability to communicate quickly so that no two developers want to work on the same piece of code

simultaneously. Strong communication and collaboration are important to ensure a centralized workflow is successful.

❖ Distributed Version Control System

Distributed version control is a form of version control in which the complete codebase, , including its full history, is mirrored on every developer's computer

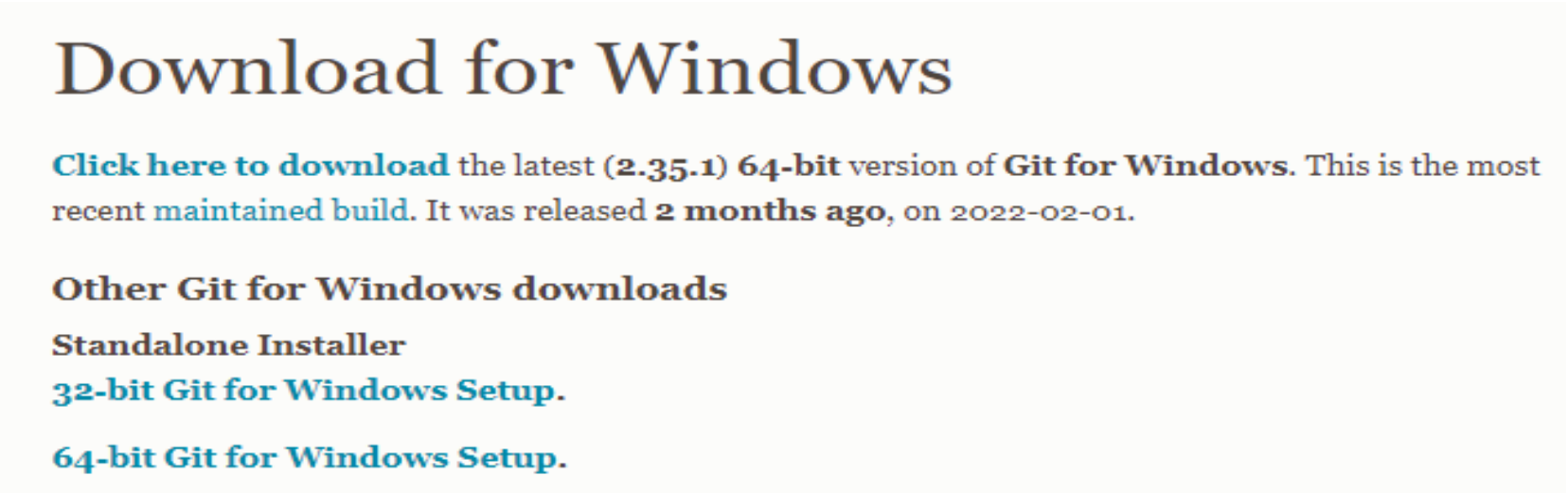
TASK 1

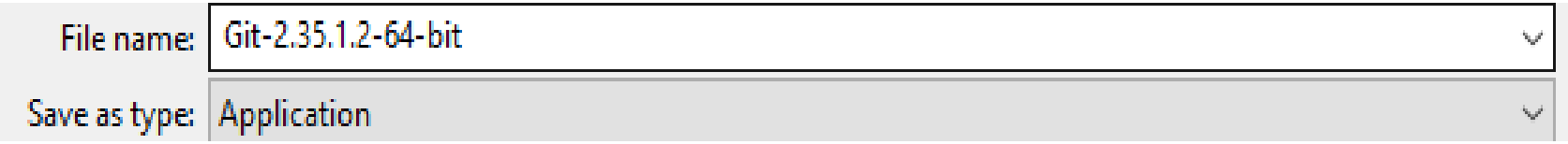
➢ Setting up Git Client

For installing git go to https://gitscm.com/download/win. The most official build is available here

1.Click the download link for windows and allow the download to complete

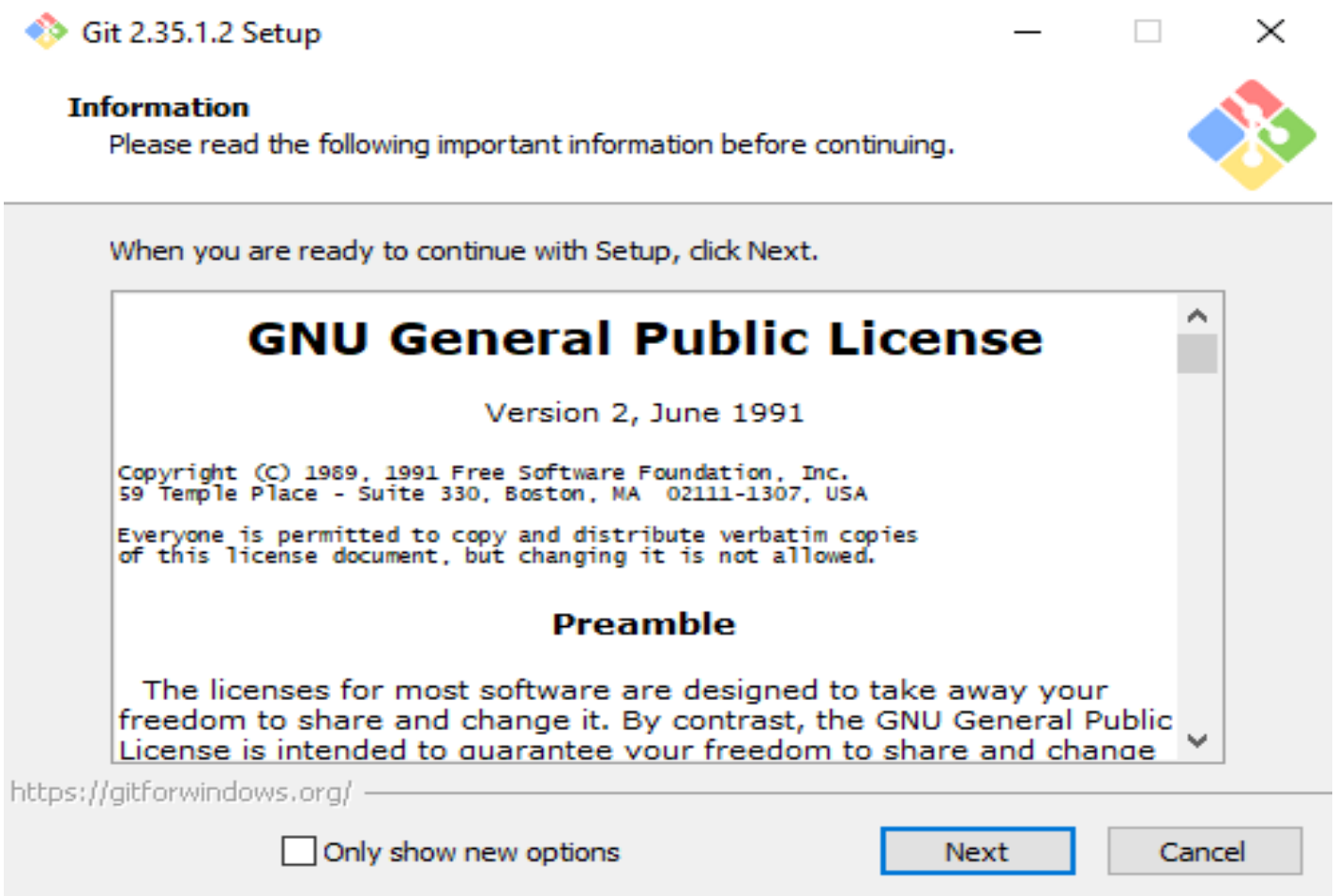
2. Select the CPU for your system now (most of the computers now have 65-bit processors)



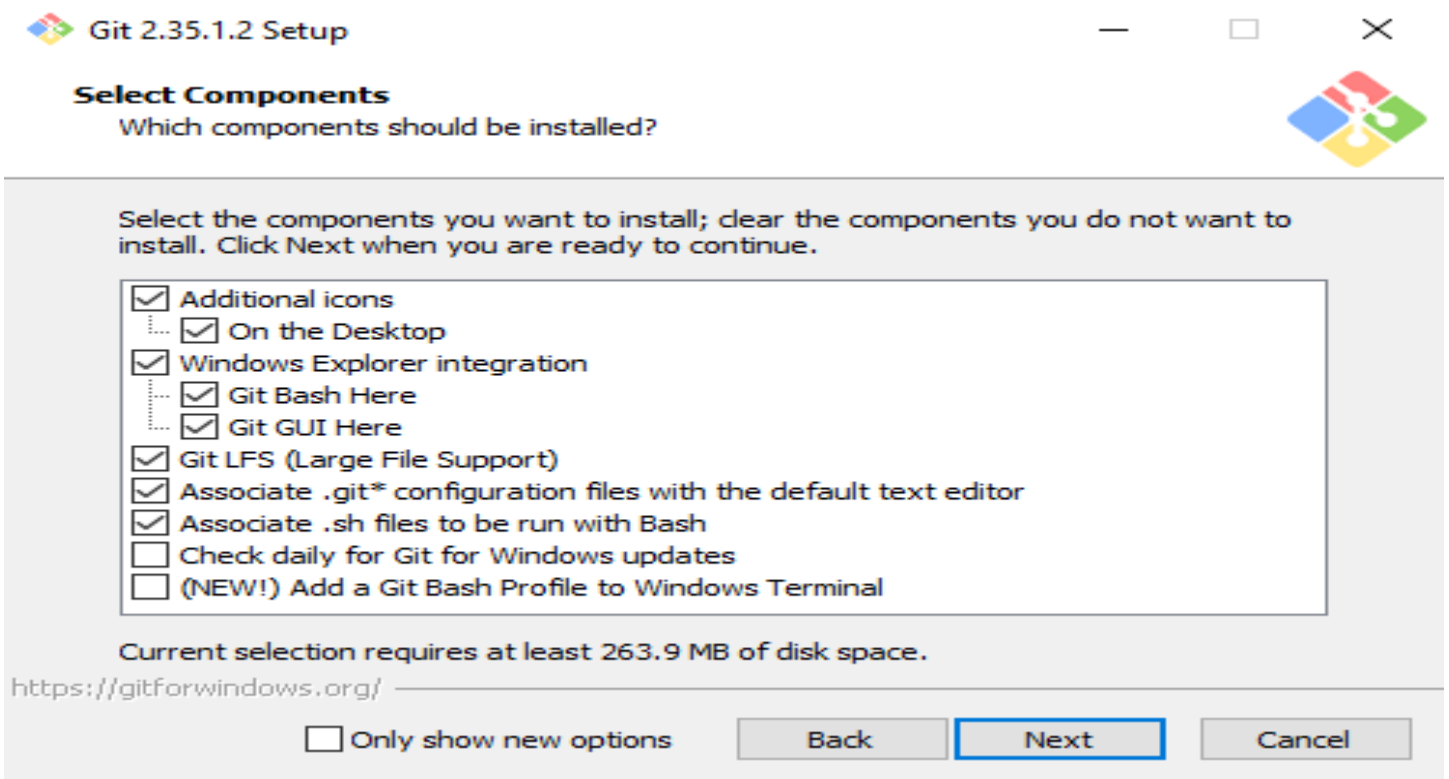


3. Once the git gets downloaded click the installer to install git

4. Review the GNU General Public License, and when you're ready to install, click Next.

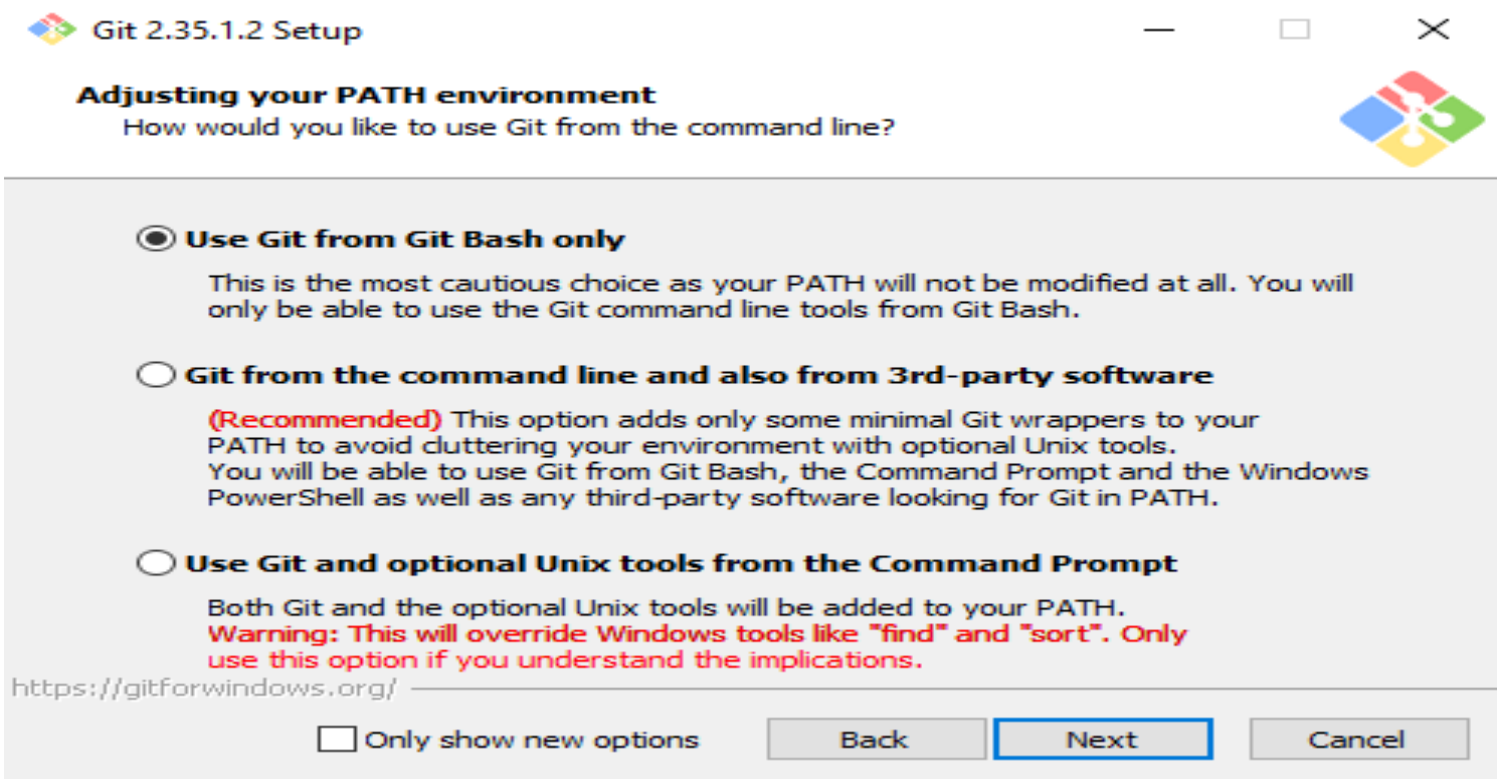


5. **A component selection screen will appear. Leave the defaults unless you have a specific need to change them and click Next.**



6. Continue clicking next for few steps

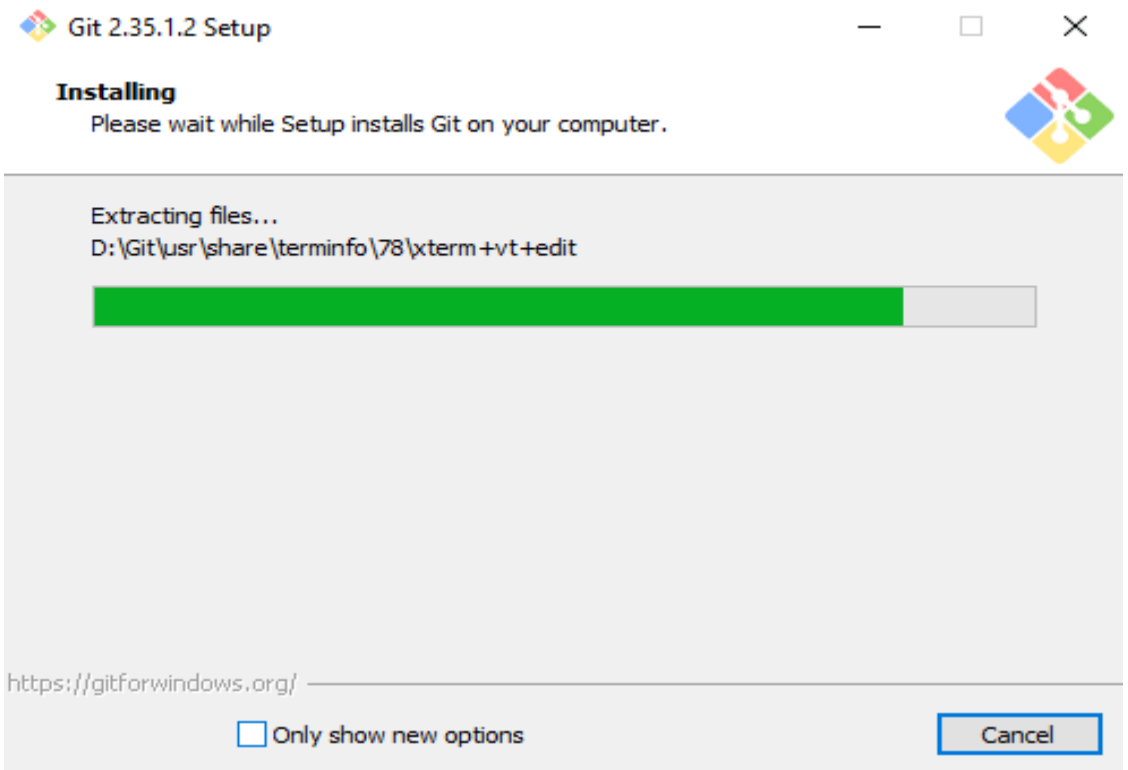
7. The next step is very important. It allows you to change the path environment



8. Continue clicking next for few more steps

9. Now click on the finish option

Now the git is installed in your local pc



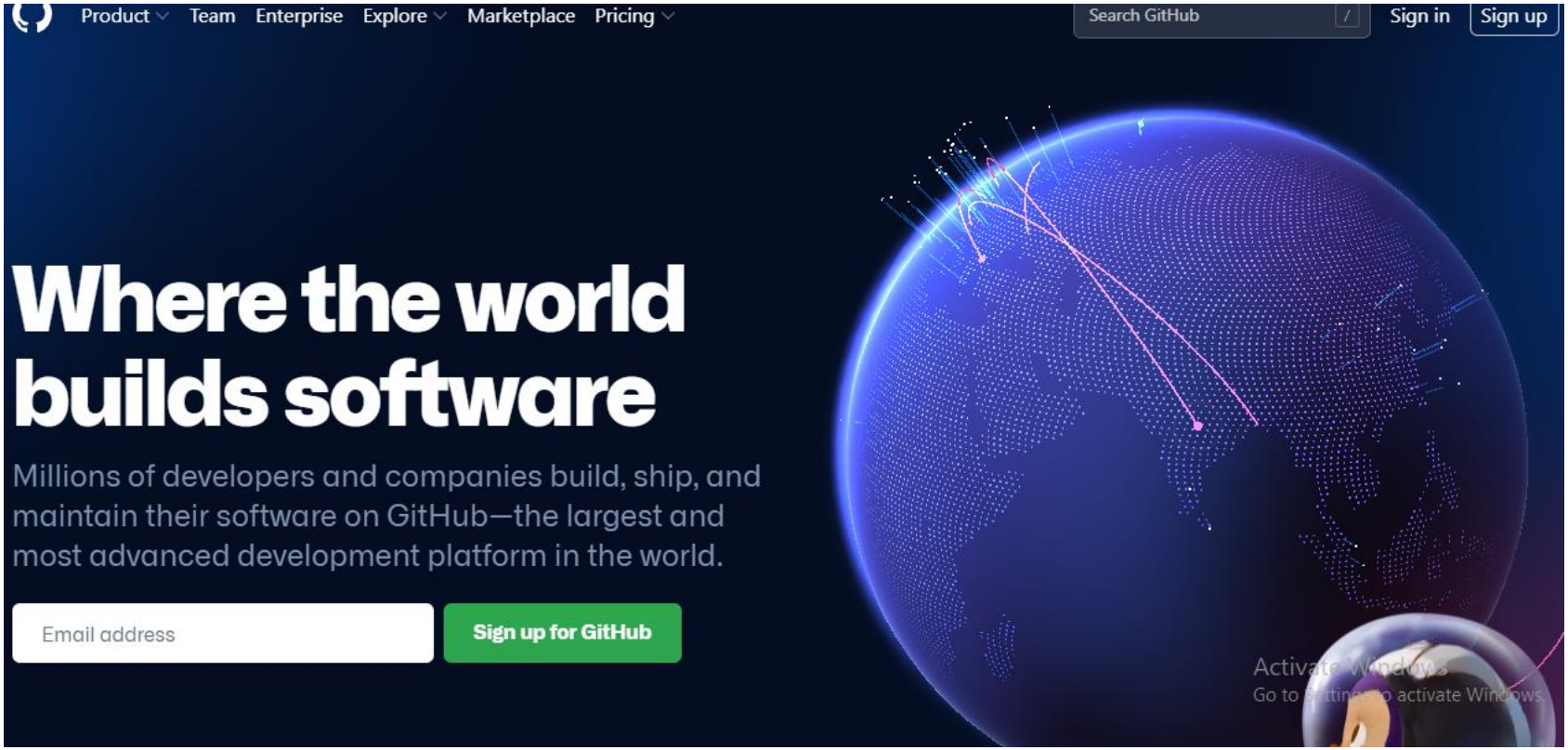
Checking Git Version

The command to check git version is :

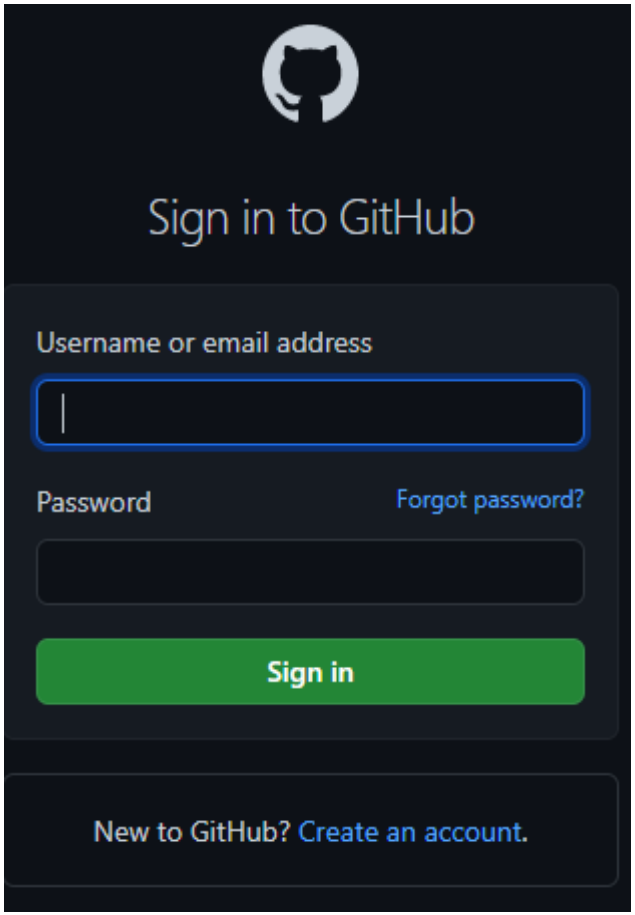
❖git --version

Aim: Setting Up GitHub Account

❖Go to the official page of GitHub. ie https://github.com.



❖If you already have an account then fill the required details. click on



sign

❖If you are new to github then click on create new account. A dialog box appears

❖Fill in the required details and your account will be created

Setting Up Username and email

```
HP@DESKTOP-D6NHS1S MINGW64 ~ (master)
$ git config --global user.name "aayush537"

HP@DESKTOP-D6NHS1S MINGW64 ~ (master)
$ git config --global user.email "aayushmittalktl2003@gmail.com"
```

Some Important Git Commands

- **git config:** It allows us to specify username and email address that will be used with our commits
- **git init:** It creates a new git repository
- **git clone:** It takes the part to the the git repository we want to clone
- **git status:** Display the status of working directoryand staging area. It let us see what changes have been staged , which files aren't be tracked by git.
- **git add:** It is used to move file from working directory to staging area
- **git commit:** It saves a log message along with a commit id of the modifications made to the git repository
- **git push :** It push the content of local repository to remote repository we have added
- **git branch:** It is used to perform operartion on the parent branch
- **git checkout :** It is used to switch to an existing branch or to create or add new branch
- **git merge :** It joins the existing branch to the main branch
- **rm –rf.git :** It removes the repository
- **touch filename:** It creates a new file in the repository

Initializing Empty Git Repository

```
HP@DESKTOP-D6NHS1S MINGW64 ~ (master)
$ mkdir scmfile
mkdir: cannot create directory 'scmfile': File exists

HP@DESKTOP-D6NHS1S MINGW64 ~ (master)
$ cd scmfile

HP@DESKTOP-D6NHS1S MINGW64 ~/scmfile (master)
$ pwd
/c/Users/HP/scmfile

HP@DESKTOP-D6NHS1S MINGW64 ~/scmfile (master)
$ vi index.py

HP@DESKTOP-D6NHS1S MINGW64 ~/scmfile (master)
$ ls
index.py

HP@DESKTOP-D6NHS1S MINGW64 ~/scmfile (master)
$ git status
On branch master

No commits yet

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

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

- **mkdir: create or make new directory**

- **git init : initialize empty git repository.**
- **Cd: change current directory**
- **Ls: list files or directory in current folder.**

Adding Files to Staging Area

```
HP@DESKTOP-D6NHS1S MINGW64 ~/scmfile (master)
$ git add index.py
warning: LF will be replaced by CRLF in index.py.
The file will have its original line endings in your working directory

HP@DESKTOP-D6NHS1S MINGW64 ~/scmfile (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   index.py
```

Commiting Files

```
HP@DESKTOP-D6NHS1S MINGW64 ~/scmfile (master)
$ git commit -m "commiting my first file index.py"
[master (root-commit) 6f3b8ea] commiting my first file index.py
 1 file changed, 5 insertions(+)
 create mode 100644 index.py
```

Generate logs

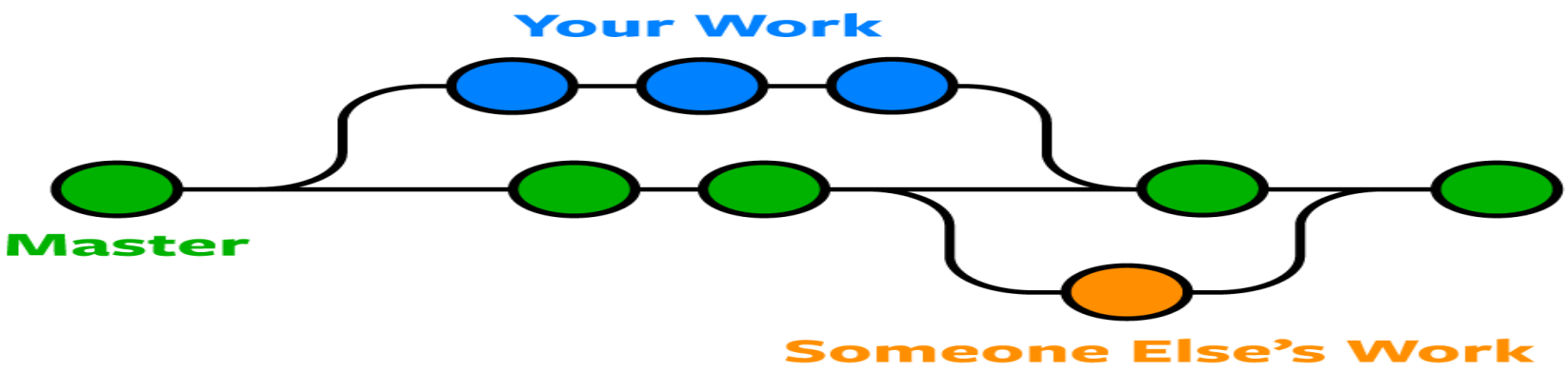
- **git log:** It shows a list of all the commands made in a repository along with a hash id. It is used for displaying the history of a repository.
- **git log –onelne:** It is used to display the output as one commit per line.

```
HP@DESKTOP-D6NHS1S MINGW64 ~/scmfile (master)
$ git log
commit 6f3b8eabb7a4ac4ac71bb1d55f56dc23685c14d9 (HEAD -> master)
Author: aayush537 <aayushmittalktl2003@gmail.com>
Date:   Mon Apr 11 23:12:49 2022 +0530

    commiting my first file index.py

HP@DESKTOP-D6NHS1S MINGW64 ~/scmfile (master)
$ git show 6f3b8eabb
commit 6f3b8eabb7a4ac4ac71bb1d55f56dc23685c14d9 (HEAD -> master)
Author: aayush537 <aayushmittalktl2003@gmail.com>
Date:   Mon Apr 11 23:12:49 2022 +0530

    commiting my first file index.py

diff --git a/index.py b/index.py
new file mode 100644
index 0000000..3a26459
--- /dev/null
+++ b/index.py
@@ -0,0 +1,5 @@
+a = 49
+b = 23
+sum = a+b
+print(sum)
+
```

Task 1.4

Git Branching:

Branching is the practice of creating copies of programs in development to work in parallel versions, retaining the original and working on the branch or making different changes to each. The default branch is the master branch. Each copy is considered a branch; the original program from which the branch is taken is referred to as the trunk.



Few Commands

- **git branch:** show all existing branches
- **git branch<branch name>:** creating new branch

- **git checkout<branch name>:** use to switch branch

```
HP@DESKTOP-D6NHS1S MINGW64 /d/array (master)
$ git branch
* master

HP@DESKTOP-D6NHS1S MINGW64 /d/array (master)
$ git branch DevA

HP@DESKTOP-D6NHS1S MINGW64 /d/array (master)
$ git branch
  DevA
* master

HP@DESKTOP-D6NHS1S MINGW64 /d/array (master)
$ git checkout DevA
Switched to branch 'DevA'

HP@DESKTOP-D6NHS1S MINGW64 /d/array (DevA)
$ git branch
* DevA
  master
```

In this we can see that the default branch is master branch which is highlighted in green. Using git branch <branchname> command we have created a new command but the default branch is still master branch using git checkout command we changed the default branch. The current branch is now DevA.

Parallel Branching

.Now let us create a new file in DevA andf compare it with the master branch

```
HP@DESKTOP-D6NHS1S MINGW64 /d/array (DevA)
$ git add -A

HP@DESKTOP-D6NHS1S MINGW64 /d/array (DevA)
$ git status
On branch DevA
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   a.txt


HP@DESKTOP-D6NHS1S MINGW64 /d/array (DevA)
$ git commit -m "commiting a.txt"
[DevA d53c9c5] commiting a.txt
 1 file changed, 1 insertion(+)
 create mode 100644 a.txt

HP@DESKTOP-D6NHS1S MINGW64 /d/array (DevA)
$ git status
On branch DevA
nothing to commit, working tree clean

HP@DESKTOP-D6NHS1S MINGW64 /d/array (DevA)
$ ls
a.txt  insertionsort.cpp  selectionsort.cpp

HP@DESKTOP-D6NHS1S MINGW64 /d/array (DevA)
$ git checkout master
Switched to branch 'master'
Your branch is up to date with 'origin/master'.

HP@DESKTOP-D6NHS1S MINGW64 /d/array (master)
$ ls
insertionsort.cpp  selectionsort.cpp
```

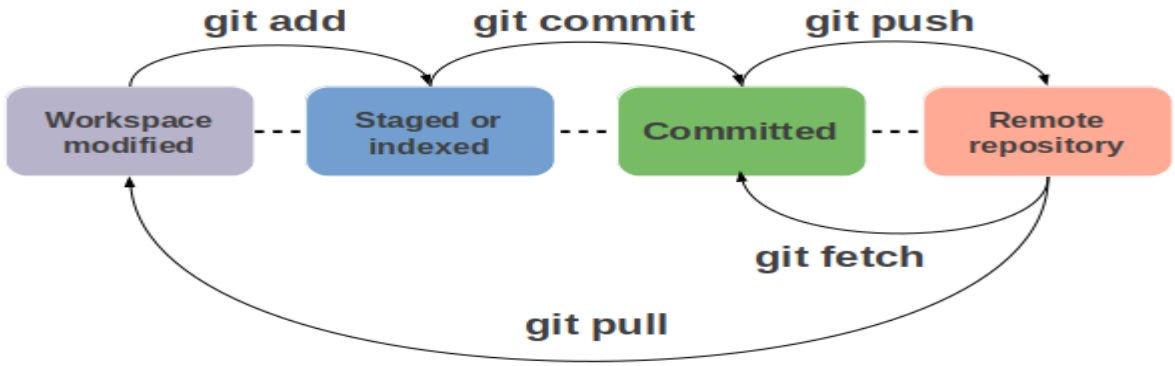
As you can see a.txt is present in DevA but not in master branch. This is how we can create parallel branches. Using git merge command we can merge DevA with master branch.

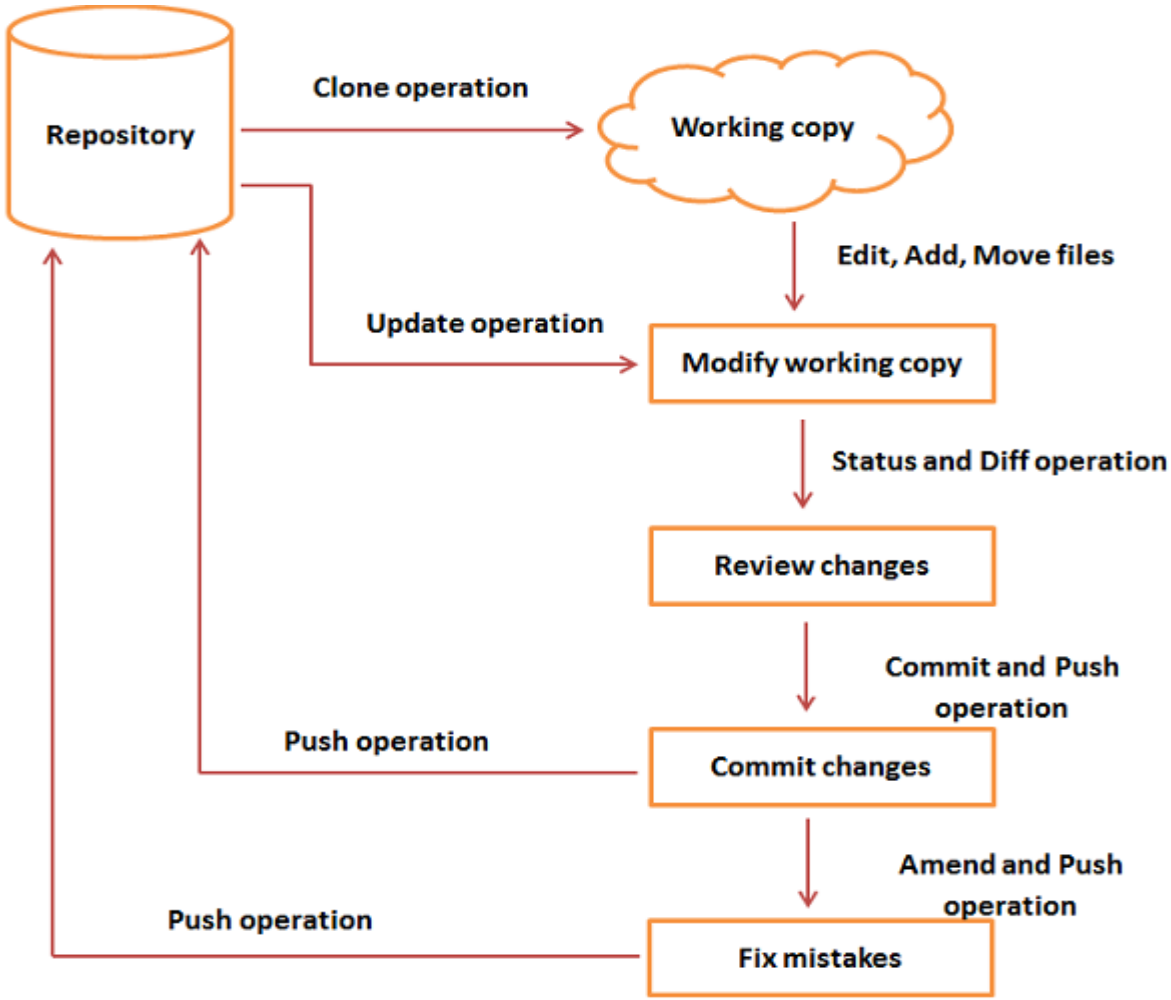
GIT LIFECYCLE DISCRIPTION

It is important to have a brief introduction about git before diving into much details. Git has three main stages that our file reside in:

1. Modified
2. Staged
3. Committed

- It's the git life-cycle





Working Directory:

It is a place where our project resides in local disk. Here the poject may or may not be tracked by git. The project can be tracked by the git using the command git init.

Staging Area:

It consists of the files which are to be a part of next commited. It is a place where different versions of our file are stored. It let git knows what changes in the fileare going to occur for the next commit. 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. The command we use to stage file is git add<filename>

Git Directory

The .git folder contains all the information that is necessary for the project and allinformation related commits, remote reository address etc. It also contains a log that stores a commit history. This log can helpyou to rollback to the desired versions of the code

TASK 1.2

Aim: Add Collaborators on Github Repository

Theory: In GitHub, we can welcome other GitHub clients to become partners in our private repositories. Being a collaborator of a personal repository you can pull content of a repository and push changes to a repository. You can include limitless 1associates public and private repositories with some each day limit restrictions.But in a private repository, the owner of arepository can only grant write access to ther collaborators, and they can't have the read only access.

Actions performed by a collaborator

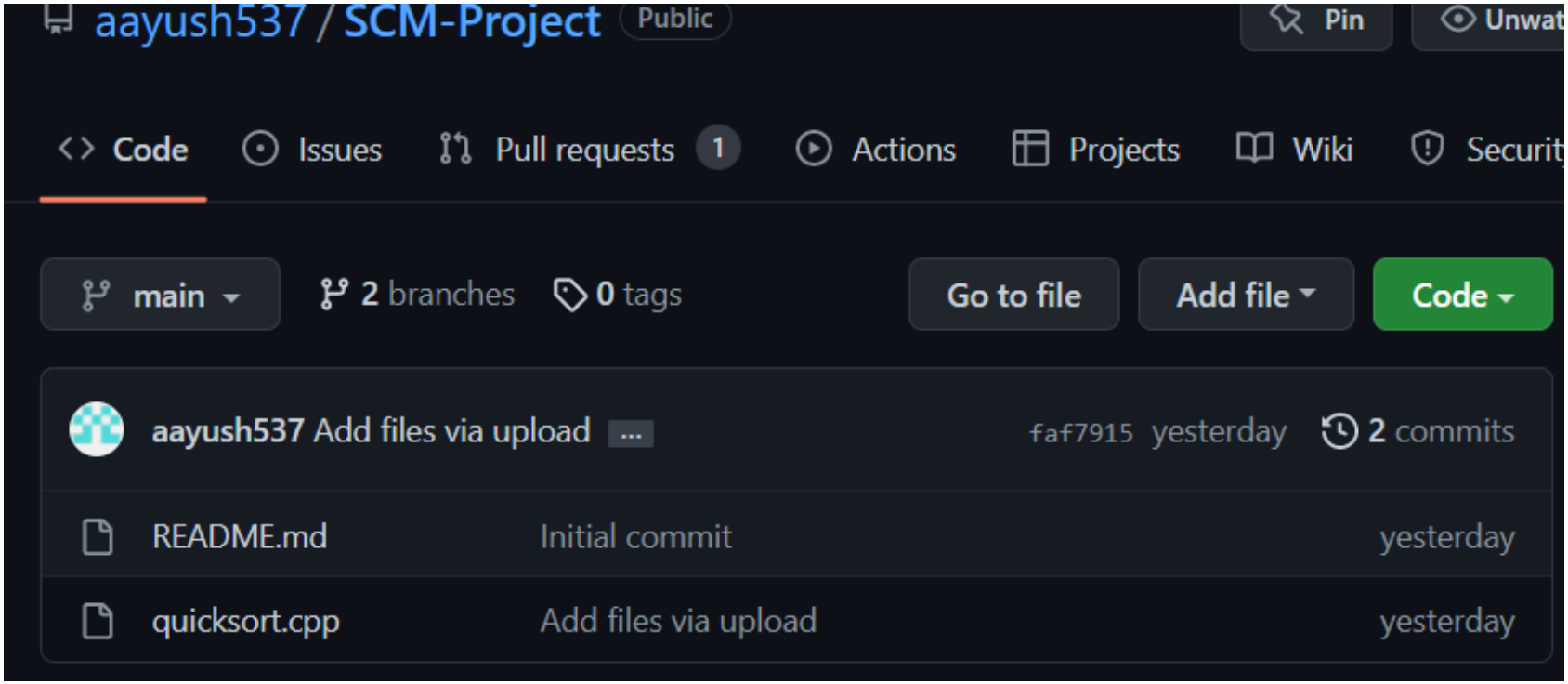
- Create, merge and close pull request in the repository.

- Fork the repositories.
- Make changes on the repositories.
- Removing themselves as collaborator on the repository.
- Mark issues or pull request as duplicate.

Steps to add collaborators

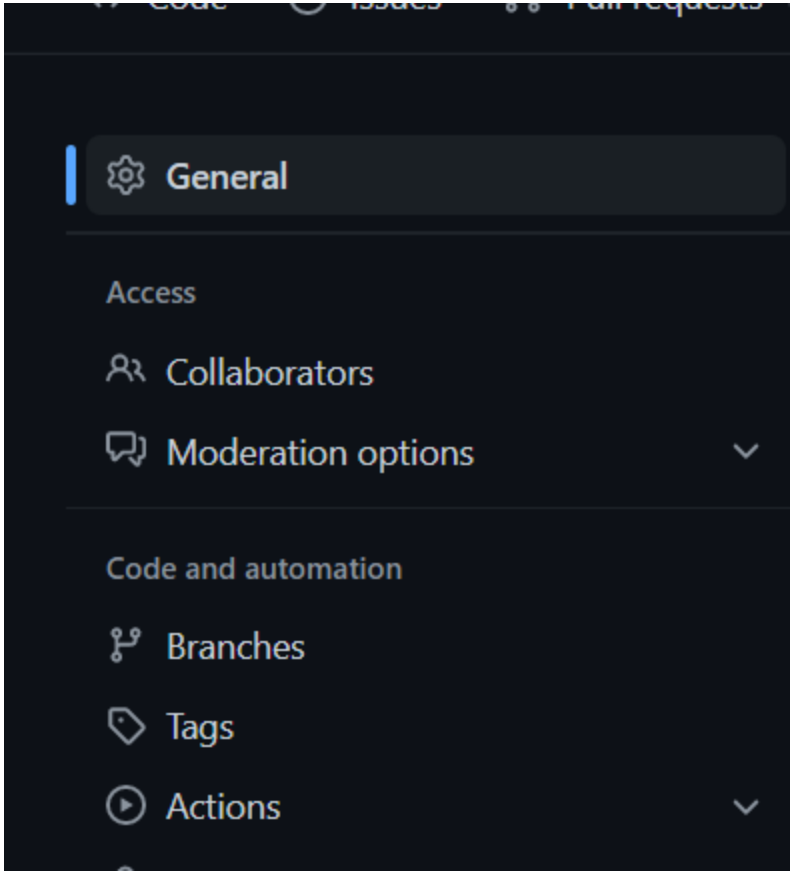
Step1:- Get the user name of the person you want to add as a collaborator

Step2: open the repository on which you want to add collaborators.



Step3: Click the settings which appears on the right hand side.

Step4: Select collaborators on left side-bar under manage access section



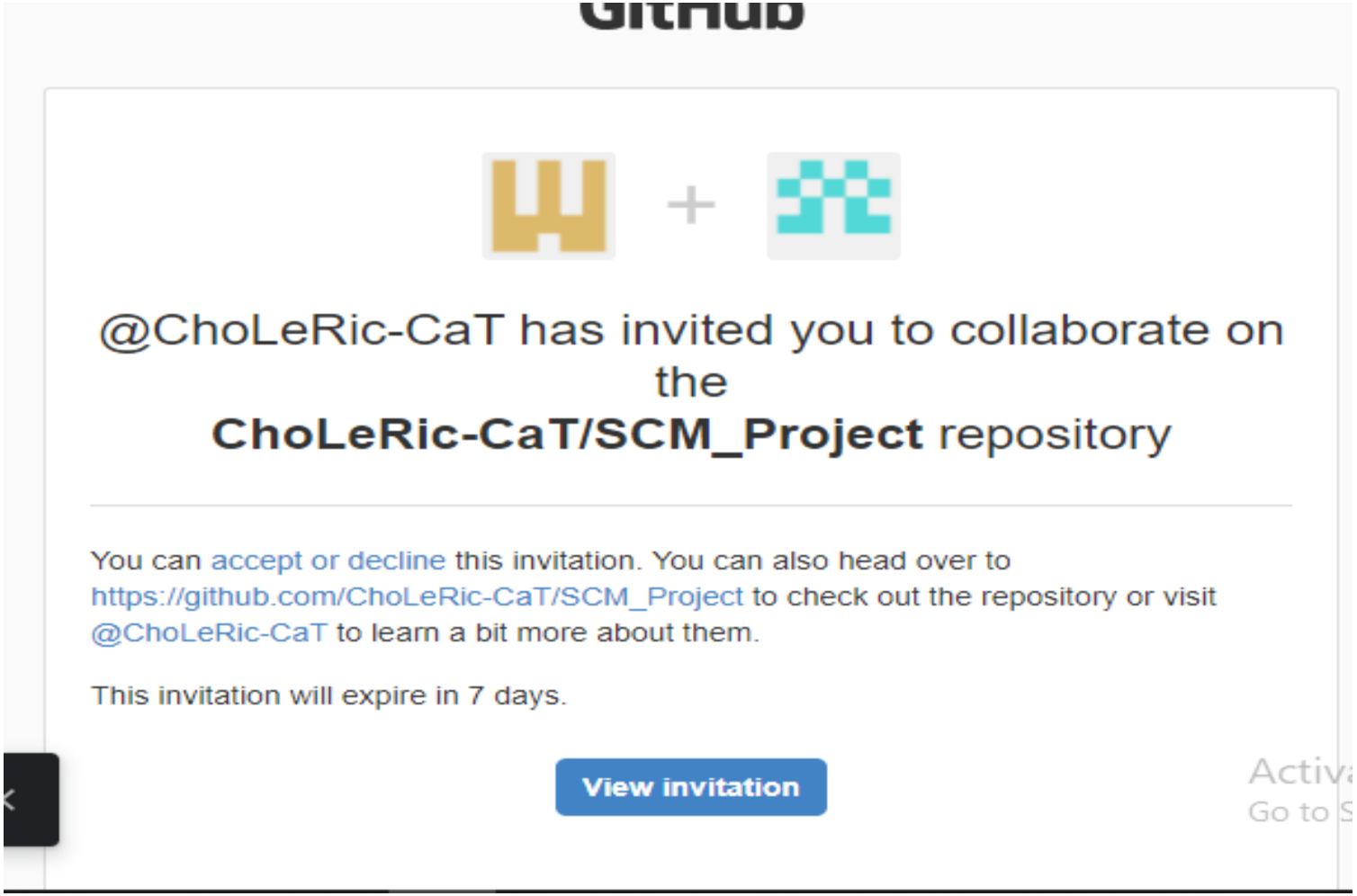
Step5: It will ask you Github Account Password saying("You are entering sudo mode")

Step6: Click on add people on right hand side.

Step7: A pop as shown below appears on the screen.

Step 8 : Search for the name of the person you wish to add as a collaborator.

Step 9: After this an invitation mail is send to the collaborator whether they want to a part of this repository or not .Once they accept the invitation they will be added as a collaborator on your repository. Till then it will be in pending invitation state



Step10 : Next the collaborator need to download a copy of owner's repository on his or her machine. This is called cloning a repo.

Removing Collaborator Permission froma person contributing to a repository.

Similar to the above steps you can go to Repository >settings> Manage access> Remove on the right side of collaborator username.

AIM: FORK AND COMMIT

About Fork

Forks are done when you want to change someone else code Fork is done on a github account. Fork creates `your own copy of repostory in a remote location(eg Github). . We can fetch updates from or submit changes to the original repository with pull request.

Fork is done for the following reasons:

1. You can freelyexperiment the changes without changing the original project.
2. This helps the maintainer of the project to better check the changes you made to the project and has the power to either accept, reject or suggest something.

Procedure:

Step1 : Go to the repository which you want to fork

Step 2: Click on the fork button on the top right corner .It will create a new fork . Add description if youwant to add, then click on create fork .

Step3:Now you have your own copy of repository.Now you can do any changes you want without modifying the original source code

Git Clone

It is use to make local copy of remote repository

Cloning a Repo in Your Device

When you create a repository on GitHub.com, it exists as a remote repository. You can clone your repository to create a local copy on your computer and sync between the two locations.

1. Once you have forked the repository, you can clone it into your computer using directly the option given on GitHub or through running git clone command in git bash.
2. Copy the URL of the forked repository

Command: git clone<repository

```
HP@DESKTOP-D6NHS1S MINGW64 /f/task (Branch01)
$ git clone https://github.com/aayush537/SCM1
Cloning into 'SCM1'...
remote: Enumerating objects: 17, done.
remote: Counting objects: 100% (17/17), done.
remote: Compressing objects: 100% (17/17), done.
remote: Total 17 (delta 2), reused 11 (delta 0), pack-reused 0
Receiving objects: 100% (17/17), 703.48 KiB | 3.82 MiB/s, done.
Resolving deltas: 100% (2/2), done.
```

url

What is Commit in Github?

Commit is like a snapshot of your repository. These commits are snapshots of your entire repository at specific times. You should make new commits often,based around logical units of change. Over time, commits should tell a story of the history of your repository and how it came to be the way that it currently is.

Commits include lots of metadata in addition to the contents and message, like the author, timestamp and more.

It is similar to saving a file that's been edited, a commit records changes to one or more files in your branch. Git assigns each commit a unique ID, called a SHA or hash,

that identifies :

- The Specific Changes
- When the Changes were made
- Who created the changes

When you make a commit, you must include a commit message that briefly describes the changes.

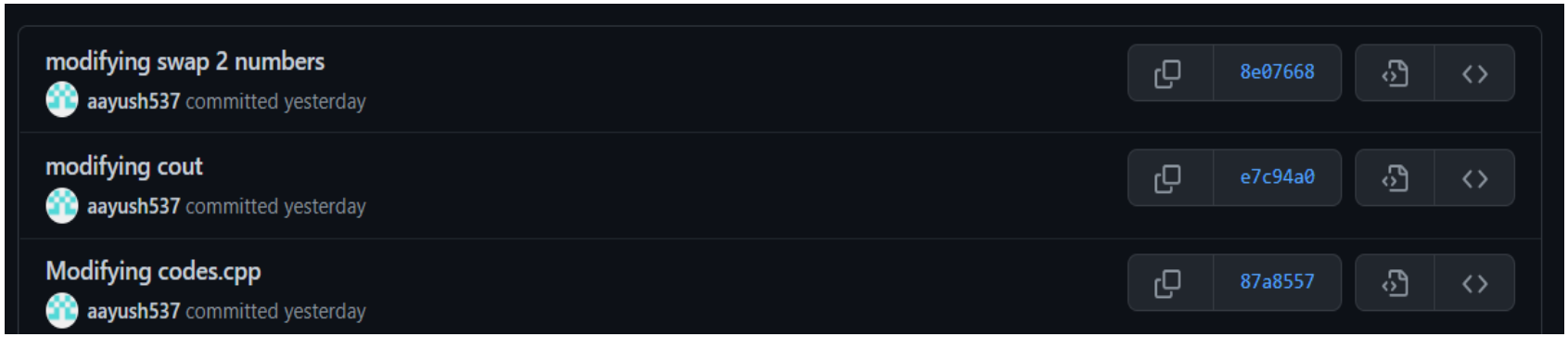
Now create a new branch and make changes in it and then commit it then push it to the Remote repository .And now the forked repository which was having only 1 branch now has 2 branches .

```
HP@DESKTOP-D6NHS1S MINGW64 /f/task (Branch01)
$ cd SCM1

HP@DESKTOP-D6NHS1S MINGW64 /f/task/SCM1 (master)
$ code.
bash: code.: command not found

HP@DESKTOP-D6NHS1S MINGW64 /f/task/SCM1 (master)
$ code .

HP@DESKTOP-D6NHS1S MINGW64 /f/task/SCM1 (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:   codes.cpp

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

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

HP@DESKTOP-D6NHS1S MINGW64 /f/task/SCM1 (master)
$ git add .
warning: LF will be replaced by CRLF in .vscode/tasks.json.
The file will have its original line endings in your working directory

HP@DESKTOP-D6NHS1S MINGW64 /f/task/SCM1 (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   .vscode/tasks.json
        modified:   codes.cpp
        new file:   codes.exe
```

```
$ git status
On branch Branch01
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:   codes.cpp
        modified:   codes.exe

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

HP@DESKTOP-D6NHS1S MINGW64 /f/task/SCM1 (Branch01)
$ git add .

HP@DESKTOP-D6NHS1S MINGW64 /f/task/SCM1 (Branch01)
$ git status
On branch Branch01
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   codes.cpp
        modified:   codes.exe


HP@DESKTOP-D6NHS1S MINGW64 /f/task/SCM1 (Branch01)
$ git commit -m "modifying cout"
[Branch01 e7c94a0] modifying cout
 2 files changed, 2 insertions(+), 1 deletion(-)

HP@DESKTOP-D6NHS1S MINGW64 /f/task/SCM1 (Branch01)
$ git status
On branch Branch01
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:   codes.cpp

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

HP@DESKTOP-D6NHS1S MINGW64 /f/task/SCM1 (Branch01)
$ git add .

HP@DESKTOP-D6NHS1S MINGW64 /f/task/SCM1 (Branch01)
$ git commit -m "modifying swap 2 numbers"
[Branch01 8e07668] modifying swap 2 numbers
 1 file changed, 8 insertions(+), 5 deletions(-)
```

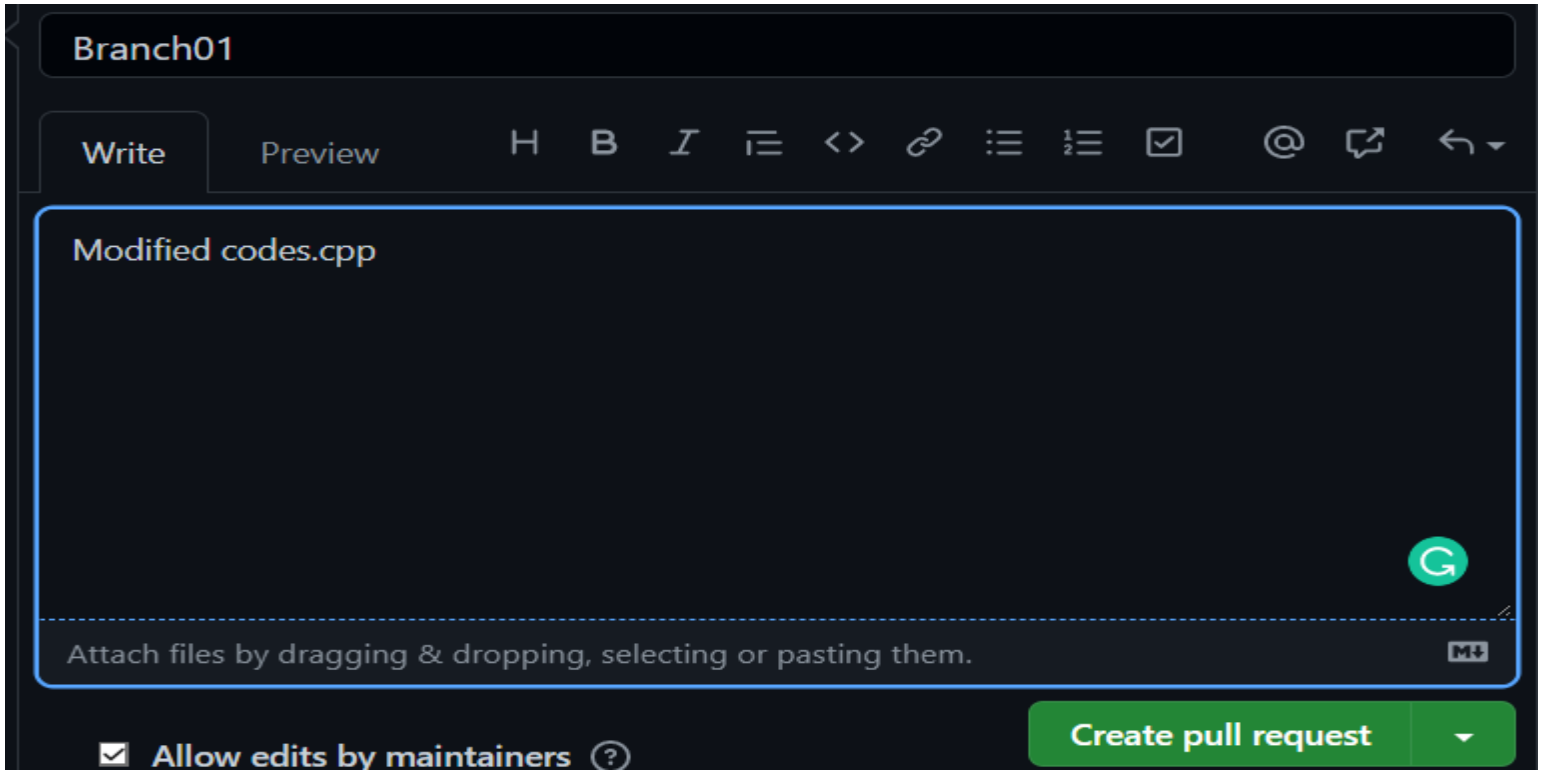
```
$ git add -A

HP@DESKTOP-D6NHS1S MINGW64 ~/Desktop/Vue/calculator-vuejs (Gatik)
$ git status
On branch Gatik
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   public/index.html
        modified:   src/assets/youtube-light.svg


HP@DESKTOP-D6NHS1S MINGW64 ~/Desktop/Vue/calculator-vuejs (Gatik)
$ git commit -m "Editing gatik's file"
[Gatik 346218b] Editing gatik's file
 2 files changed, 2 insertions(+), 2 deletions(-)

HP@DESKTOP-D6NHS1S MINGW64 ~/Desktop/Vue/calculator-vuejs (Gatik)
$ git push -u origin Gatik
Enumerating objects: 27, done.
Counting objects: 100% (27/27), done.
Delta compression using up to 4 threads
Compressing objects: 100% (19/19), done.
Writing objects: 100% (19/19), 1.83 KiB | 469.00 KiB/s, done.
Total 19 (delta 11), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (11/11), completed with 6 local objects.
remote:
remote: Create a pull request for 'Gatik' on GitHub by visiting:
remote:      https://github.com/aayush537/calculator-vuejs/pull/new/Gatik
remote:
To https://github.com/aayush537/calculator-vuejs.git
 * [new branch]      Gatik -> Gatik
branch 'Gatik' set up to track 'origin/Gatik'.
```

❖ You commits are reflected in the forked repository on github



.

❖ Now you can send pull request to the owner of the repository.



❖Now the forked repository have 2 branches.

MERGE AND RESOLVE CONFLICTS CREATED DUE TO OWN ACTIVITY AN COLLABORATORS ACTIVITY.

What is a merge conflict?

- A merge conflict is an event that will take place when git is unable to resolve differences in code between the two commits automatically.
- Gi t can merge the changes only if the commits are o different lines or branches

In our repository we and our collaborators make changes and if changes are made in the same file on a particular line in two separate branches then a merge conflict will arise. Then we have to finalize the best version of our code and commit itc

Resolve conficts created due to own activity

1. Create a file and make some chamges and push it to the github repository.

```
HP@DESKTOP-D6NHS1S MINGW64 ~ (master)
$ cd Desktop/quickSort

HP@DESKTOP-D6NHS1S MINGW64 ~/Desktop/quickSort (main)
$ git init
Reinitialized existing Git repository in C:/Users/HP/Desktop/quickSort/.git/

HP@DESKTOP-D6NHS1S MINGW64 ~/Desktop/quickSort (main)
$ code .

HP@DESKTOP-D6NHS1S MINGW64 ~/Desktop/quickSort (main)
$ git commit -a -m "modifying function"
[main 21ecfa2] modifying function
 1 file changed, 3 insertions(+), 3 deletions(-)

HP@DESKTOP-D6NHS1S MINGW64 ~/Desktop/quickSort (main)
$ git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean
```

```
HP@DESKTOP-D6NHS1S MINGW64 ~/Desktop/quickSort (main)
$ git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

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:   quicksort.cpp

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

HP@DESKTOP-D6NHS1S MINGW64 ~/Desktop/quickSort (main)
$ git add .

HP@DESKTOP-D6NHS1S MINGW64 ~/Desktop/quickSort (main)
$ git checkout -m "commiting main"
error: pathspec 'commiting main' did not match any file(s) known to git

HP@DESKTOP-D6NHS1S MINGW64 ~/Desktop/quickSort (main)
$ git commit -m "Commiting main"
[main 78bcca4] Commiting main
 1 file changed, 10 insertions(+), 4 deletions(-)
```

```
HP@DESKTOP-D6NHS1S MINGW64 ~/Desktop/quickSort (main)
$ git push origin main
Enumerating objects: 8, done.
Counting objects: 100% (8/8), done.
Delta compression using up to 4 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (6/6), 770 bytes | 770.00 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/aayush537/SCM-Project.git
   faf7915..78bcca4  main -> main
```

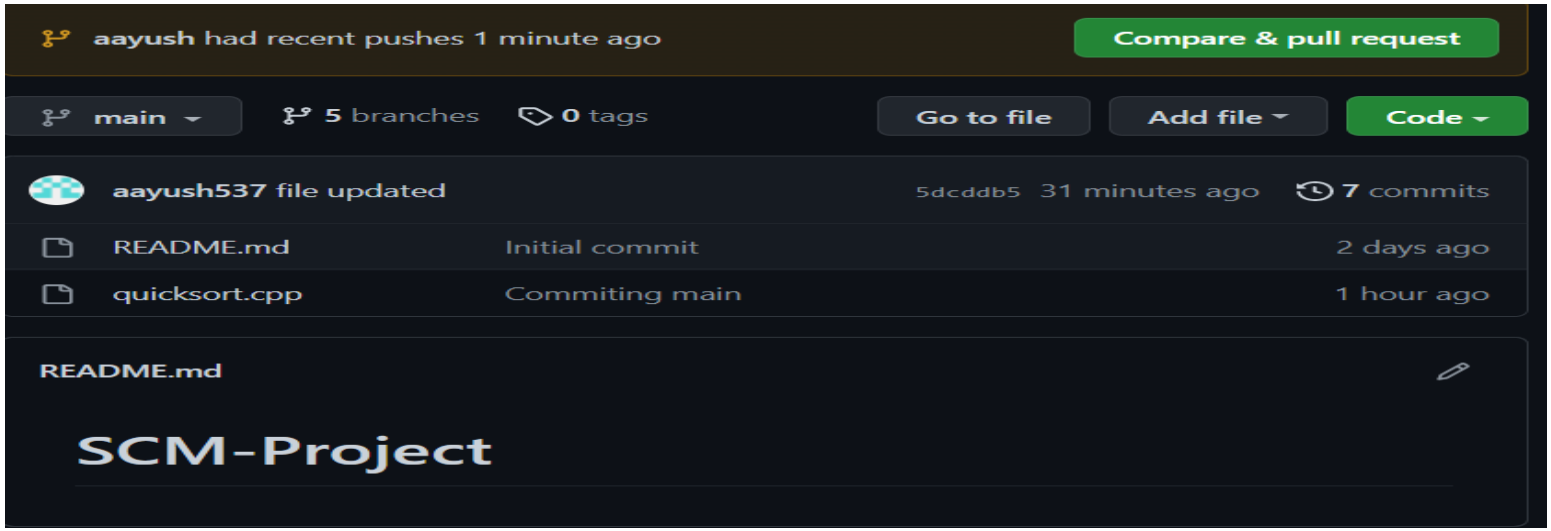
- Create a new branch , make some changes in it and the most important thing to note is to make changes in the same line so as to create conflicts and commit and push this as well. Now your repo will have 2 branches.
- Below you can see the whole process from creating a branch to pushing on GitHub:

```
HP@DESKTOP-D6NHS1S MINGW64 ~/Desktop/quickSort (main)
$ git checkout -b aayush
Switched to a new branch 'aayush'

HP@DESKTOP-D6NHS1S MINGW64 ~/Desktop/quickSort (aayush)
$ git commit -a -m "Modifying same lines"
[aayush e5285a1] Modifying same lines
 1 file changed, 3 insertions(+), 3 deletions(-)

HP@DESKTOP-D6NHS1S MINGW64 ~/Desktop/quickSort (aayush)
$ git status
On branch aayush
nothing to commit, working tree clean

HP@DESKTOP-D6NHS1S MINGW64 ~/Desktop/quickSort (aayush)
$ git checkout master
Switched to branch 'master'
Your branch is up to date with 'origin/master'.
```

```
HP@DESKTOP-D6NHS1S MINGW64 ~/Desktop/quickSort (aayush)
$ git add .

HP@DESKTOP-D6NHS1S MINGW64 ~/Desktop/quickSort (aayush)
$ git add .

HP@DESKTOP-D6NHS1S MINGW64 ~/Desktop/quickSort (aayush)
$ git commit -m "removing error"
[aayush 2c306de] removing error
 1 file changed, 3 insertions(+), 3 deletions(-)
```

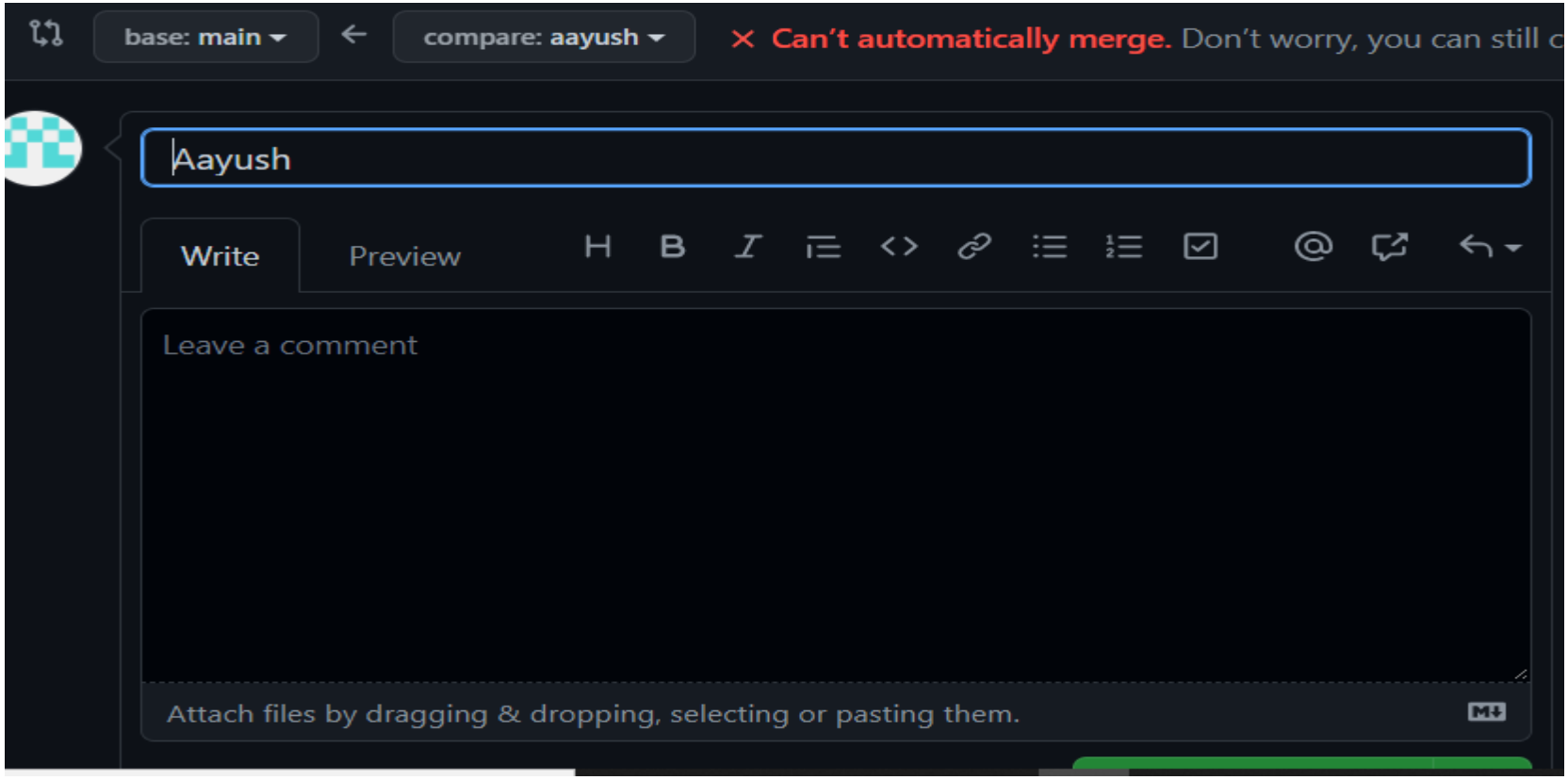
```
HP@DESKTOP-D6NHS1S MINGW64 ~/Desktop/quickSort (aayush)
$ git push origin aayush
Enumerating objects: 8, done.
Counting objects: 100% (8/8), done.
Delta compression using up to 4 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (6/6), 667 bytes | 667.00 KiB/s, done.
Total 6 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 1 local object.
remote:
remote: Create a pull request for 'aayush' on GitHub by visiting:
remote:      https://github.com/aayush537/SCM-Project/pull/new/aayush
remote:
To https://github.com/aayush537/SCM-Project.git
 * [new branch]      aayush -> aayush
```

- On your Github desktop repo you'll be able to see the changes you made after switching to the aayush branch



- Now when you try to merge commits made in branch aayush with the commits made in branch main you will get conflicts which need to be resolved

```
HP@DESKTOP-D6NHS1S MINGW64 ~/Desktop/quickSort (main)
$ git merge aayush
Auto-merging quicksort.cpp
CONFLICT (content): Merge conflict in quicksort.cpp
Automatic merge failed; fix conflicts and then commit the result.
```



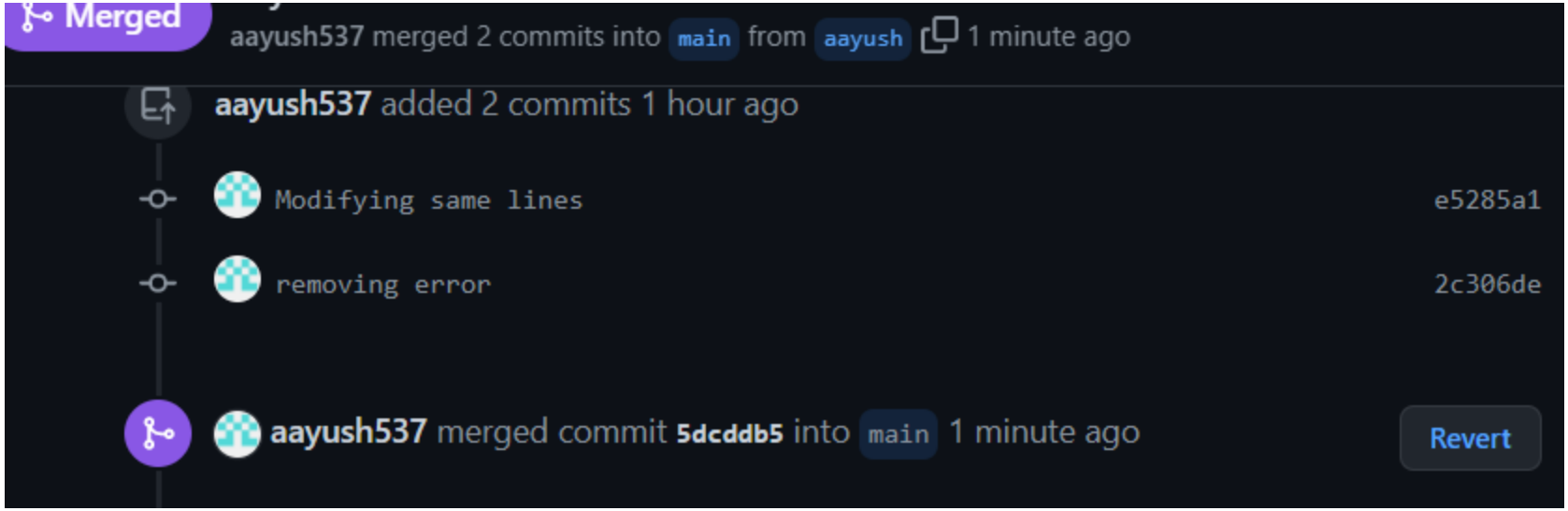
➢ **Now we use git mergetool command to merge the conflicts manually**

```
HP@DESKTOP-D6NHS1S MINGW64 ~/Desktop/quickSort (main|MERGING)
$ git mergetool

This message is displayed because 'merge.tool' is not configured.
See 'git mergetool --tool-help' or 'git help config' for more details.
'git mergetool' will now attempt to use one of the following tools:
opendiff kdiff3 tkdiff xxdiff meld tortoisemerge gvimdiff diffuse diffmerge ecme
rge p4merge araxis bc codecompare smerge emerge vimdiff nvimdiff
Merging:
quicksort.cpp

Normal merge conflict for 'quicksort.cpp':
  {local}: modified file
  {remote}: modified file
Hit return to start merge resolution tool (vimdiff):
4 files to edit
```

- After this a dashboard like this will appear

```
#include<iostream>
using namespace std;
void swap(int arr[], int i, int j){
                int temp = arr[i];
                arr[i]=arr[j];
                arr[j]= temp;
            }

int partition(int arr[], int l, int r){
    int pivot=arr[r];
    int i = l-1;
    for(int j=l; j<r; j++){
+ +-- 10 lines: if(arr[j]<pivot){----------
        int pi = partition(arr, l, r);
        quickSort(arr, l, pi-1);
        quickSort(arr, pi+1, r);
    }
}
int main(){
<<<<<<< HEAD
    int n;
    cin>>n;
    int arr[n];
    for(int k=0; k<n; k++){
        cin>>arr[k];
    }

    quickSort(arr, 0, n-1);
    for(int i=0; i<n ; i++){
=======
    int arr[5]={5, 4, 3, 2,1};
    quickSort(arr, 0, 4);
    for(int i=0; i<5; i++){
>>>>>>> aayush
        cout<<arr[i]<<" ";

    }cout<<endl;
```


➢ **After resolving all the conflicts a dashboard like this appears**



RESET AND REVERT

While working on a version control system , It is unvoidable that we need to roolback certain changes either dueto bug or temp code revert. There are three ways by which we can undo our commits .

These are reset revert and checkout. Git checkout and git revertt in fact can be used to manipulate commits or individual files. However git reset –hard command is quite distructive.

Let's see all these commands one by one

1. git checkout -- <file name>=> **This command is used to remove all unstaged changes i.e the changes that are present in our working directory.**

1 Create an empty folder.

2 Now add a file to it

```
HP@DESKTOP-D6NHS1S MINGW64 ~/Desktop (master)
$ mkdir test

HP@DESKTOP-D6NHS1S MINGW64 ~/Desktop (master)
$ cd test

HP@DESKTOP-D6NHS1S MINGW64 ~/Desktop/test (master)
$ git init
Initialized empty Git repository in C:/Users/HP/Desktop/test/.git/

HP@DESKTOP-D6NHS1S MINGW64 ~/Desktop/test (master)
$ git status
On branch master

No commits yet

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

```python
hungry = input("Are you hungry")
if hungry =="yes":
    print("eat samosas")
    print("eat pizza")
else:
    print("Do you work")
```

```
HP@DESKTOP-D6NHS1S MINGW64 ~/Desktop/test (master)
$ git checkout -- hungry.py

HP@DESKTOP-D6NHS1S MINGW64 ~/Desktop/test (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:   hungry.py
```

s

```
hungry = input("Are you hungry")
if hungry =="yes":
    print("eat samosas")

else:
    print("Do you work")
```

GIT REVERT

This command is used for undoing changes to remote repository. The git revert command reverts the changes introduced by the commit and appends a new commit with resulting reversed content.

We made a new file and committed changeand run the git revert command .

```
HP@DESKTOP-D6NHS1S MINGW64 ~/Desktop (master)
$ mkdir test

HP@DESKTOP-D6NHS1S MINGW64 ~/Desktop (master)
$ cd test

HP@DESKTOP-D6NHS1S MINGW64 ~/Desktop/test (master)
$ git init
Initialized empty Git repository in C:/Users/HP/Desktop/test/.git/

HP@DESKTOP-D6NHS1S MINGW64 ~/Desktop/test (master)
$ git status
On branch master

No commits yet

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

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

HP@DESKTOP-D6NHS1S MINGW64 ~/Desktop/test (master)
$ git add hungry.py

HP@DESKTOP-D6NHS1S MINGW64 ~/Desktop/test (master)
$ git commit "Creating hungry.py"
error: pathspec 'Creating hungry.py' did not match any file(s) known to git

HP@DESKTOP-D6NHS1S MINGW64 ~/Desktop/test (master)
$ git commit -m "Creating hungry.py"
[master (root-commit) 31cda0f] Creating hungry.py
 1 file changed, 6 insertions(+)
 create mode 100644 hungry.py
```

```
HP@DESKTOP-D6NHS1S MINGW64 ~/Desktop/test (master)
$ git commit -am "burger"
[master f0b5b91] burger
 1 file changed, 2 insertions(+), 1 deletion(-)

HP@DESKTOP-D6NHS1S MINGW64 ~/Desktop/test (master)
$ git log
commit f0b5b9158bb07621b7d756eaaddc71aa969da190 (HEAD -> master)
Author: aayush <aayushmittalktl2003@gmail.com>
Date:   Sat Jun 4 19:51:02 2022 +0530

    burger

commit feb78a01bf3b738d3464a188cf040f3806133e27
Author: aayush <aayushmittalktl2003@gmail.com>
Date:   Sat Jun 4 19:49:06 2022 +0530

    pizza

commit 31cda0f7beb86b37f8926781d611379e3a8cd644
Author: aayush <aayushmittalktl2003@gmail.com>
Date:   Sat Jun 4 19:41:27 2022 +0530

    Creating hungry.py

HP@DESKTOP-D6NHS1S MINGW64 ~/Desktop/test (master)
$ git revert  f0b5b9158bb07621b7d756eaaddc71aa969da190
[master d8027db] Revert "burger"
 1 file changed, 1 insertion(+), 2 deletions(-)

HP@DESKTOP-D6NHS1S MINGW64 ~/Desktop/test (master)
$ git log
commit d8027db13e6d7dd4aac77e12ca9430fbc53f05dd (HEAD -> master)
Author: aayush <aayushmittalktl2003@gmail.com>
Date:   Sat Jun 4 19:52:23 2022 +0530

    Revert "burger"

    This reverts commit f0b5b9158bb07621b7d756eaaddc71aa969da190.

commit f0b5b9158bb07621b7d756eaaddc71aa969da190
Author: aayush <aayushmittalktl2003@gmail.com>
Date:   Sat Jun 4 19:51:02 2022 +0530

    burger

commit feb78a01bf3b738d3464a188cf040f3806133e27
Author: aayush <aayushmittalktl2003@gmail.com>
Date:   Sat Jun 4 19:49:06 2022 +0530
```

Git revert –n command

The command used for it is : git revert –n <commit id>: It is used for undoing changes to repository's commit history. The git revert command reverts the changes introduced by the commit and push it to unstaged area

```
HP@DESKTOP-D6NHS1S MINGW64 ~/Desktop/test (master)
$ git revert -n feb78a01bf3b738d3464a188cf040f3806133e27
error: Your local changes to the following files would be overwritten by merge:
        hungry.py
Please commit your changes or stash them before you merge.
Aborting
fatal: revert failed

HP@DESKTOP-D6NHS1S MINGW64 ~/Desktop/test (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:   hungry.py
```

```
Revert "burger"

This reverts commit fe342c8c7caef94d136cd347b60f15188a2faac6.

# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
# On branch master
# Changes to be committed:
#       modified:   hungry.py
#
~
~
~
~
~
~
~
```

What is git reset command?

The git reset command is used to move the current head to the commit specified. It will undo all the the commits after the specified commit. It deletes the commit only from the local repository not from the remote repository

Git reset –soft Head~1 – This command removes the commit but does not unstage the file . Our changes are still present in staging area.

git reset -- mixed Head~1=> -This is the default command which removes the commit as well as unstages the file and our changes are stored inthe working directory.

Git reset – hard : It completely destroy any change and remove them from local repository

```
HP@DESKTOP-D6NHS1S MINGW64 ~/Desktop/test (master)
$ git reset --hard f0b5b9158bb07621b7d756eaaddc71aa969da190
HEAD is now at f0b5b91 burger

HP@DESKTOP-D6NHS1S MINGW64 ~/Desktop/test (master)
$ git log
commit f0b5b9158bb07621b7d756eaaddc71aa969da190 (HEAD -> master)
Author: aayush <aayushmittalktl2003@gmail.com>
Date:   Sat Jun 4 19:51:02 2022 +0530

    burger

commit feb78a01bf3b738d3464a188cf040f3806133e27
Author: aayush <aayushmittalktl2003@gmail.com>
Date:   Sat Jun 4 19:49:06 2022 +0530

    pizza

commit 31cda0f7beb86b37f8926781d611379e3a8cd644
Author: aayush <aayushmittalktl2003@gmail.com>
Date:   Sat Jun 4 19:41:27 2022 +0530

    Creating hungry.py
```

```
hungry = input("Are you hungry")
if hungry =="yes":
    print("eat samosas")
    print("eat pizza")
    print("Eat burger")
else:
    print("Do you work")
```

Project

Project With Team : Demstrating all aspects of Git

According to the given Task, each member has created a distributed repository and added team members, several task was done accordingly.

Opened and closed a pull request, each member created a pull request on other team member's repository and closed the pull request generated by other team members

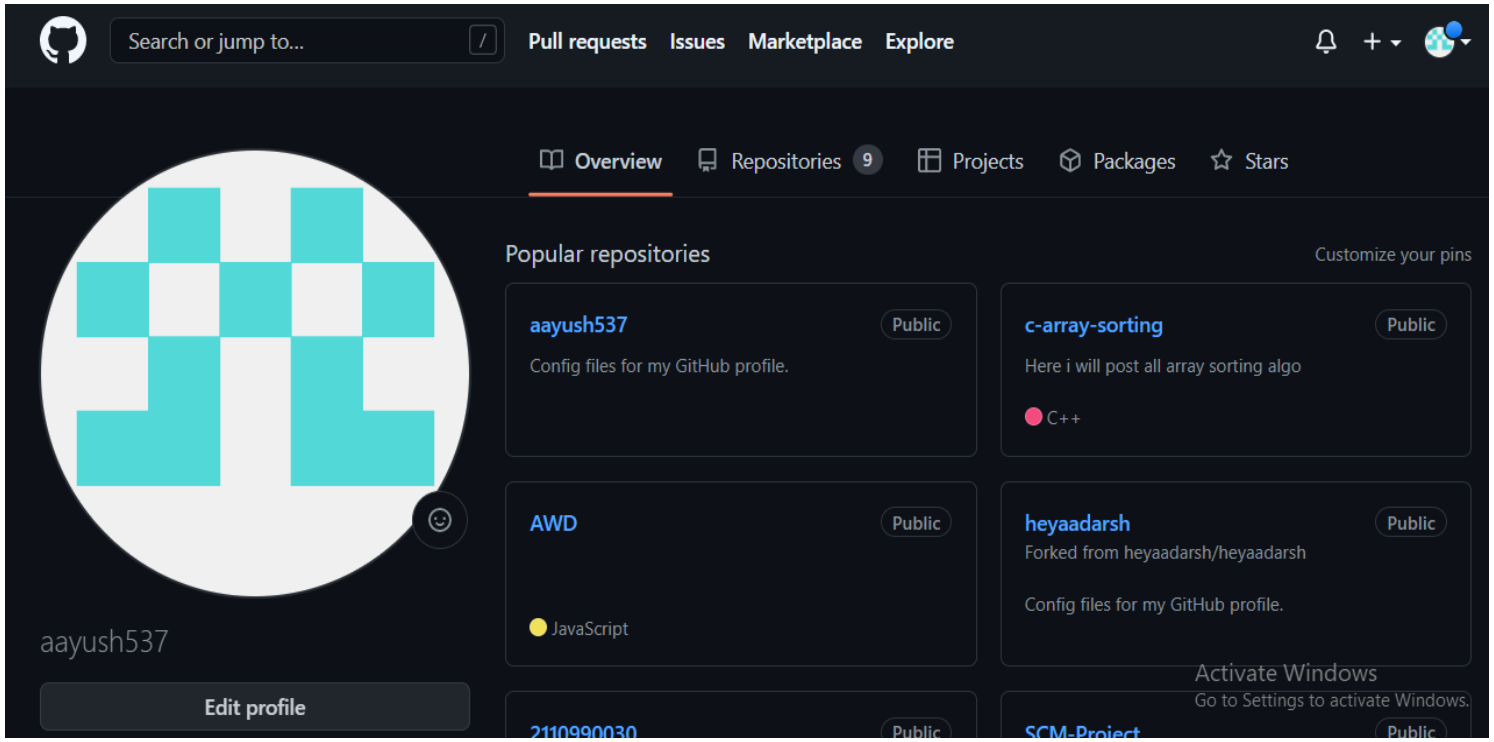
on the respective repository as being a maintainer. Later network graph was published.

Team Members

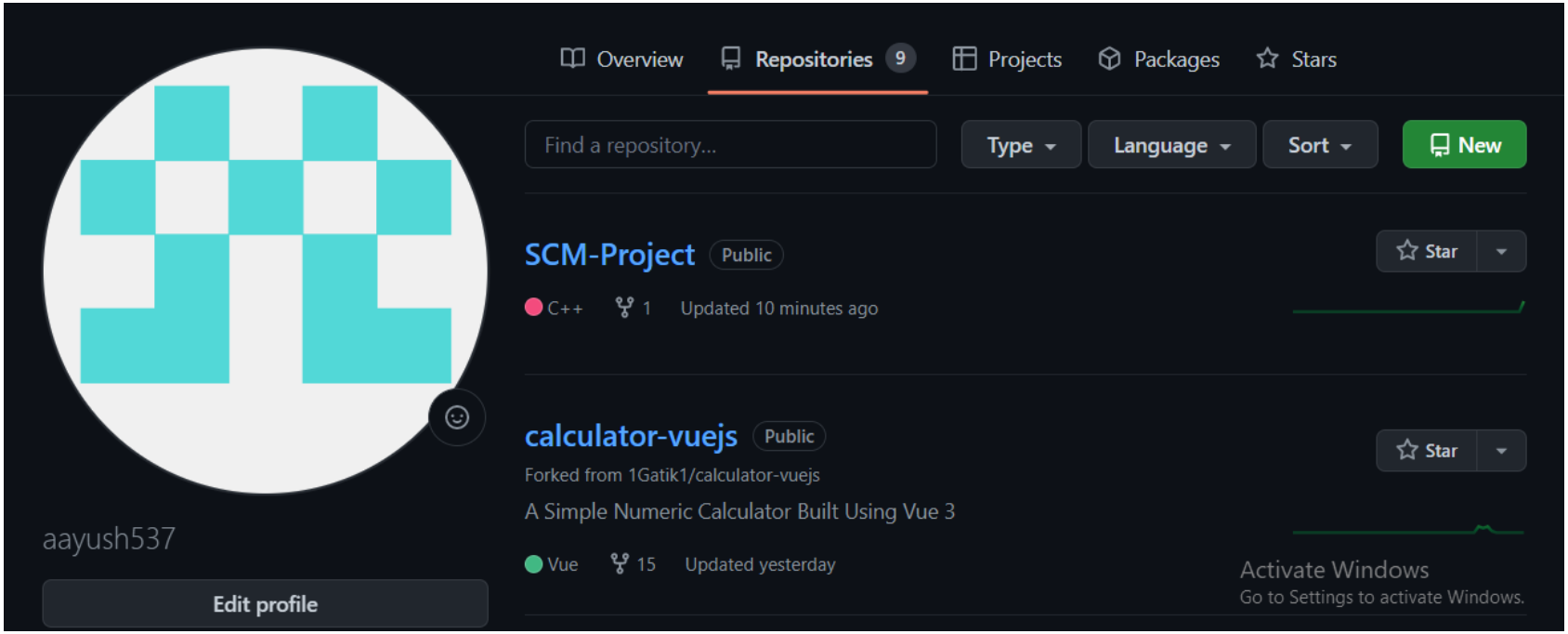
- Aayush Mittal 2110990030
- Abhinn Singh Bisht 2110990030
- Ankusha Sabharwal 2110990209
- Dhruv Jain 2110990443
- Gatik Veer 2110990493

Aim: Creating a distributed repository and add members in a Project Team

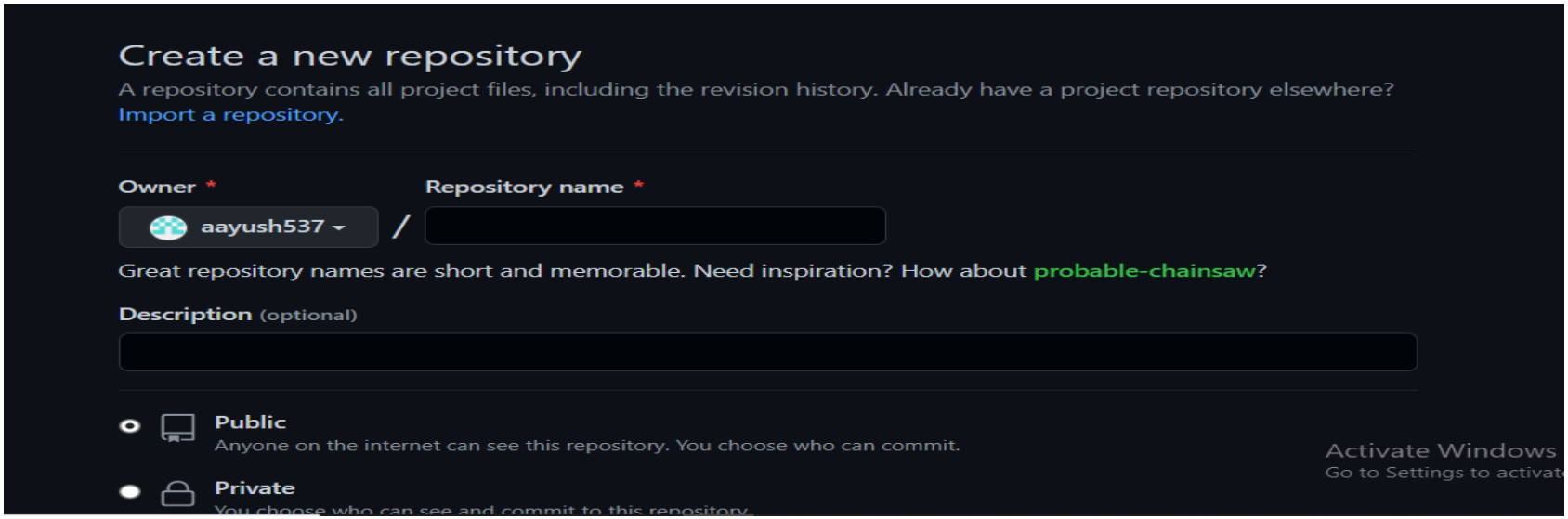
- Login to your Github account and you will land on the homepage as shown below. Click on Repositories option in the menu bar.



- Click on the 'New' button on the top right corner.



- Enter the repository name and add a short description about it if you want.



- Click on "Create Repository".
- Now, you have created your repository successfully.

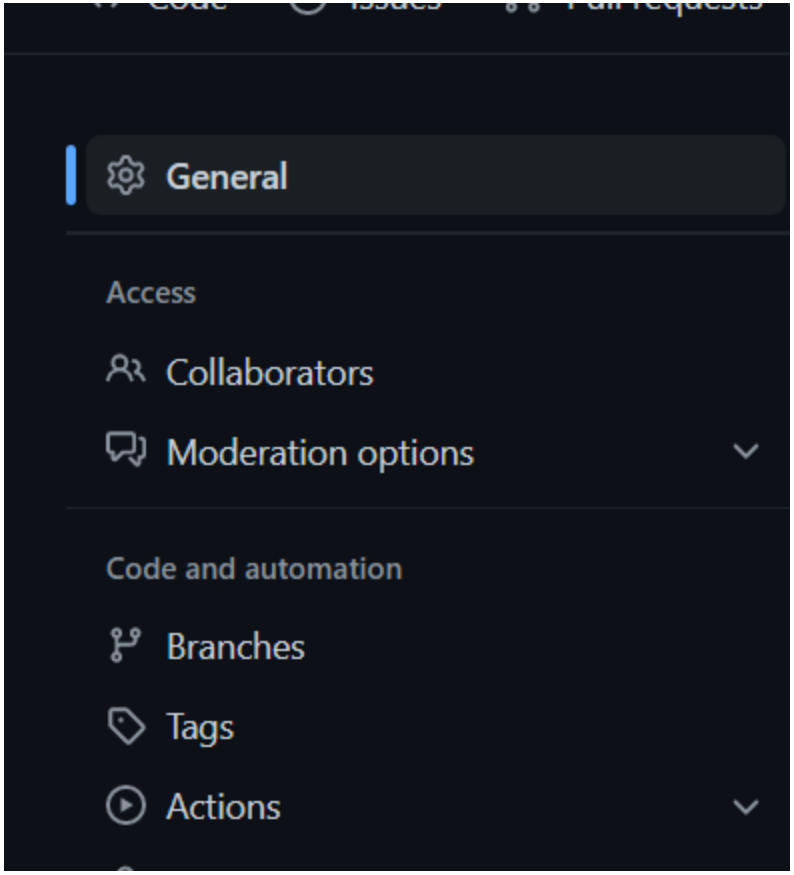
Adding Collaborators on your repository

Step1:- Get the user name of the person you want to add as a collaborator

Step2: Open the repository on which you want to add collaborators.

Step3: Click the settings which appears on the right hand side.

Step4: Select collaborators on left side-bar under manage access section

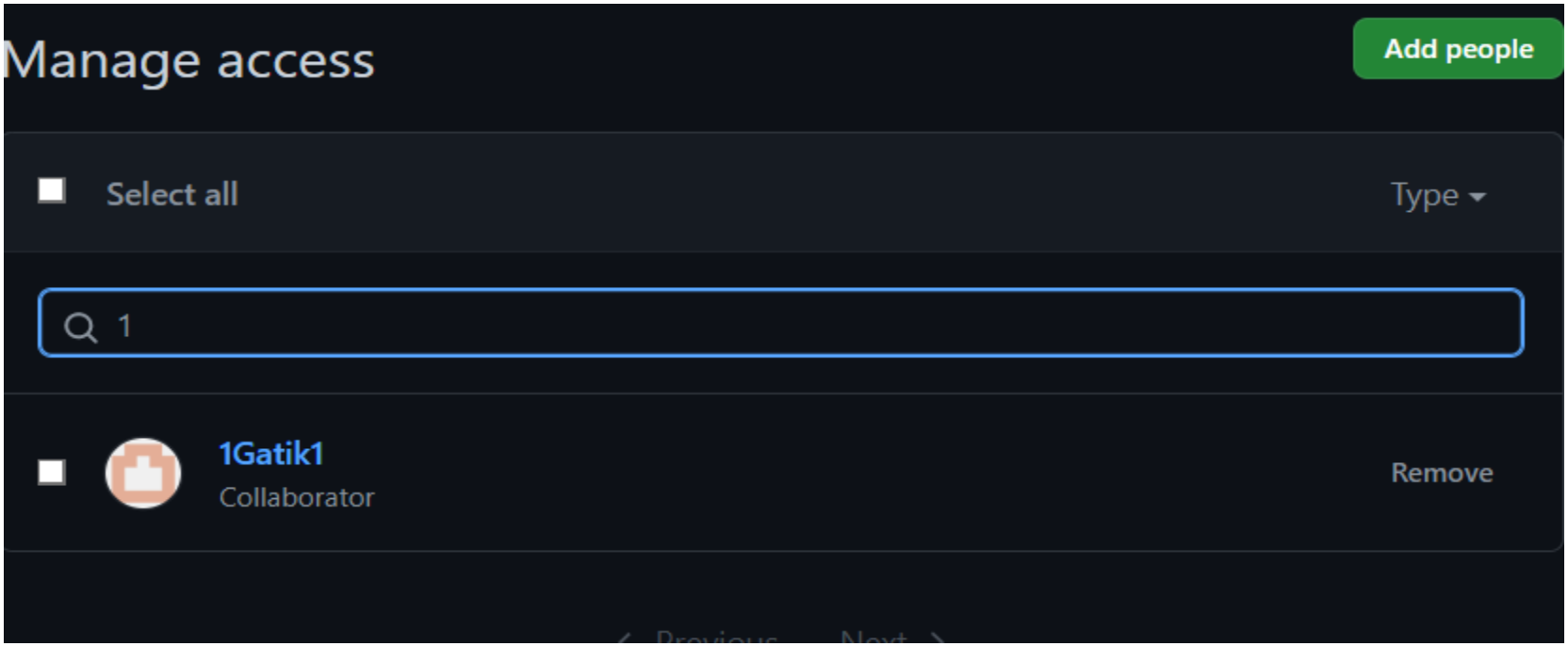


Step5: It will ask you Github Account Password saying("You are entering sudo mode")

Step6: Click on add people on right hand side.

Step7: A pop as shown below appears on the screen.

Step 8 : Search for the name of the person you wish to add as a collaborator.



Step 9: After this an invitation mail is send to the collaborator whether they want to a part of this repository or not .Once they accept the invitation they will be added as a collaborator on your repository. Till then it will be in pending invitation state. Now you have done adding a single collaborator.

GitHub

@aayush537 has invited you to collaborate on the **aayush537/SCM-Project** repository

You can accept or decline this invitation. You can also head over to https://github.com/aayush537/SCM-Project to check out the repository or visit @aayush537 to learn a bit more about them.

This invitation will expire in 7 days.

View invitation

Note: This invitation was intended for **gatik0493.be21@chitkara.edu.in**. If you were not expecting this invitation, you can ignore this email. If @aayush537 is sending you too many emails, you can block them or report abuse.

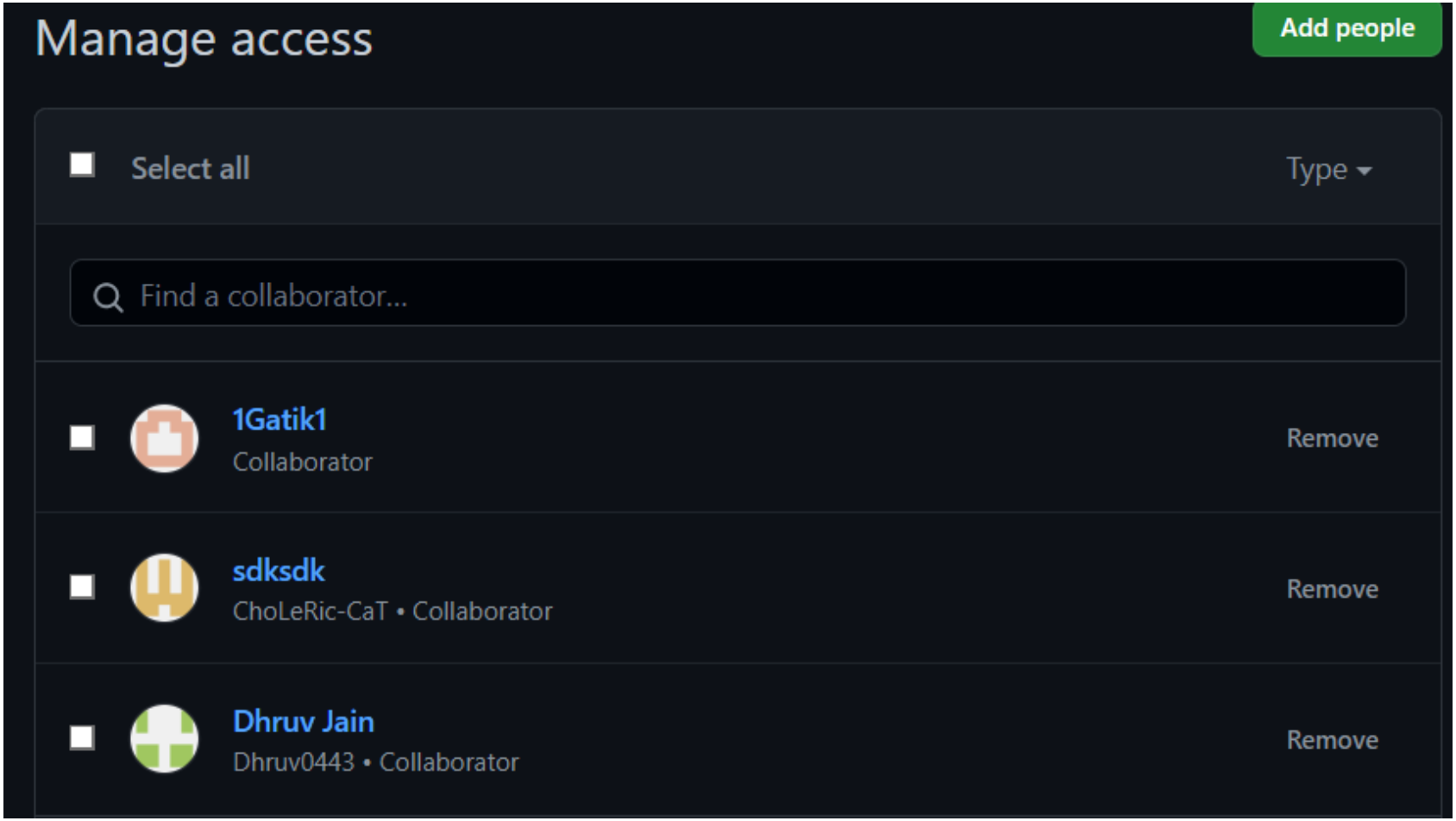
Getting a 404 error? Make sure you're signed in as **1Gatik1**.

Likewise all other collaborators need to be added.

Step10 : Next the collaborator need to download a copy of owner's repository on his or her machine. This is called cloning a repo.

Removing Collaboration Permission Contributing to a Repository

Similar to the above steps, go to Your Repository -> Settings -> Manage Access -> Remove (on the right side of collaborator username)

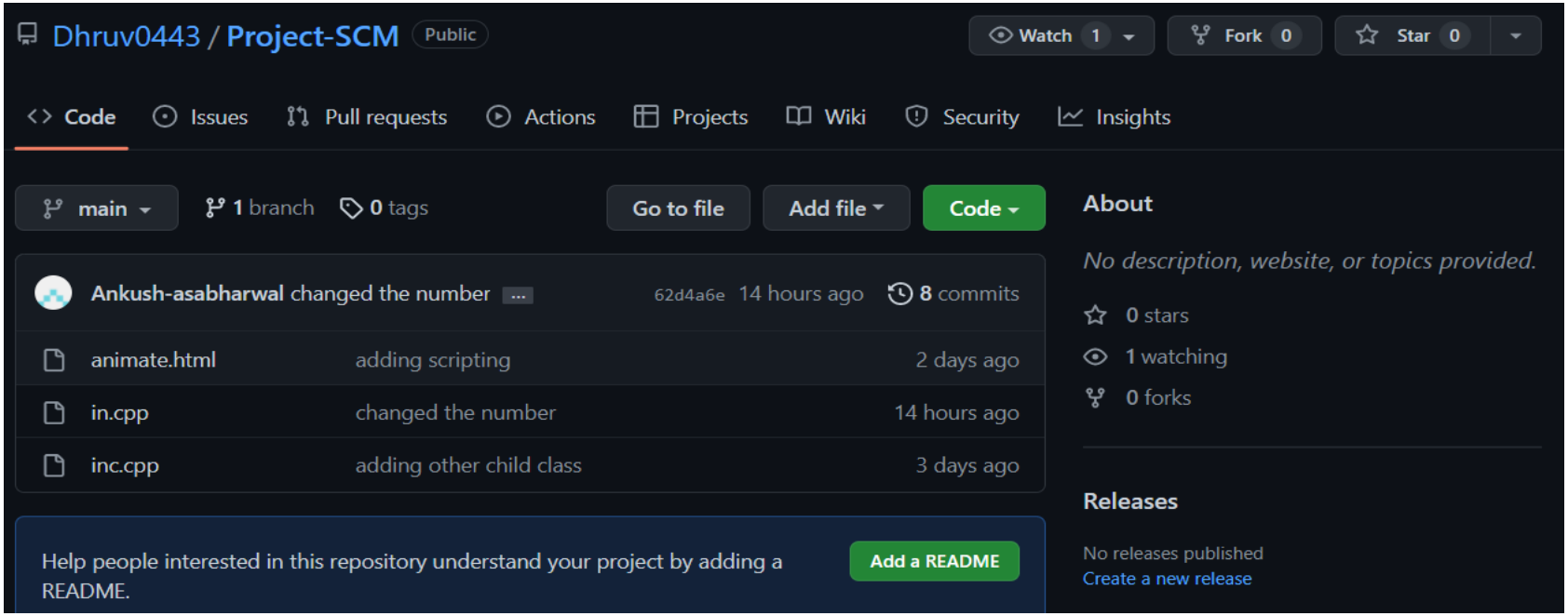


Aim : Open and close pull request

For opening and closing a pull request we have to undergo the following steps:

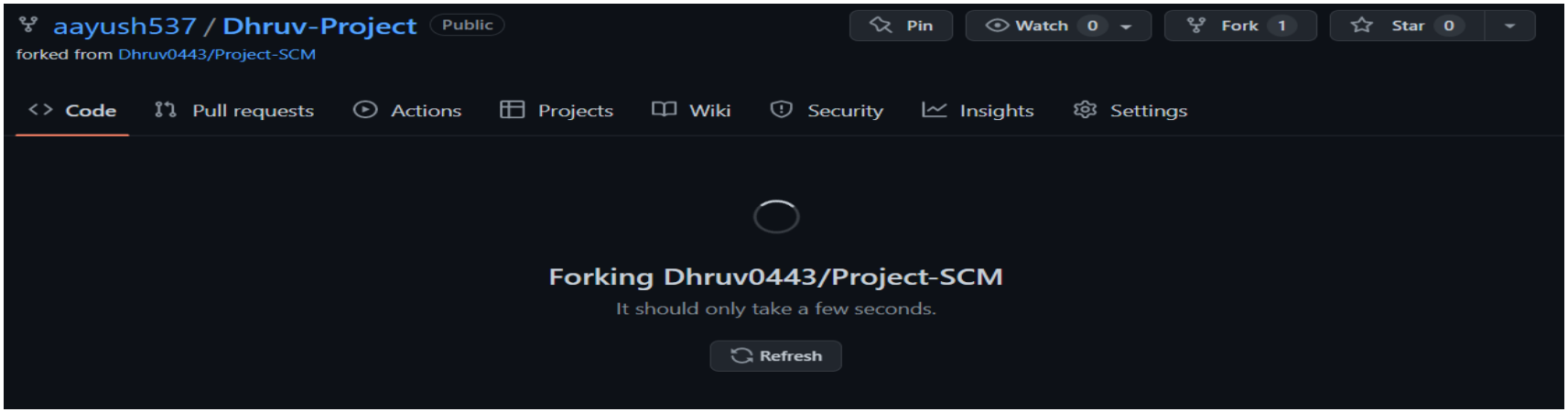
- Fork someone's repository
- Clone it to your local repository
- Commit some changes
- Push it to the remote repository
- Make pull request

Step 1: Fork a repository

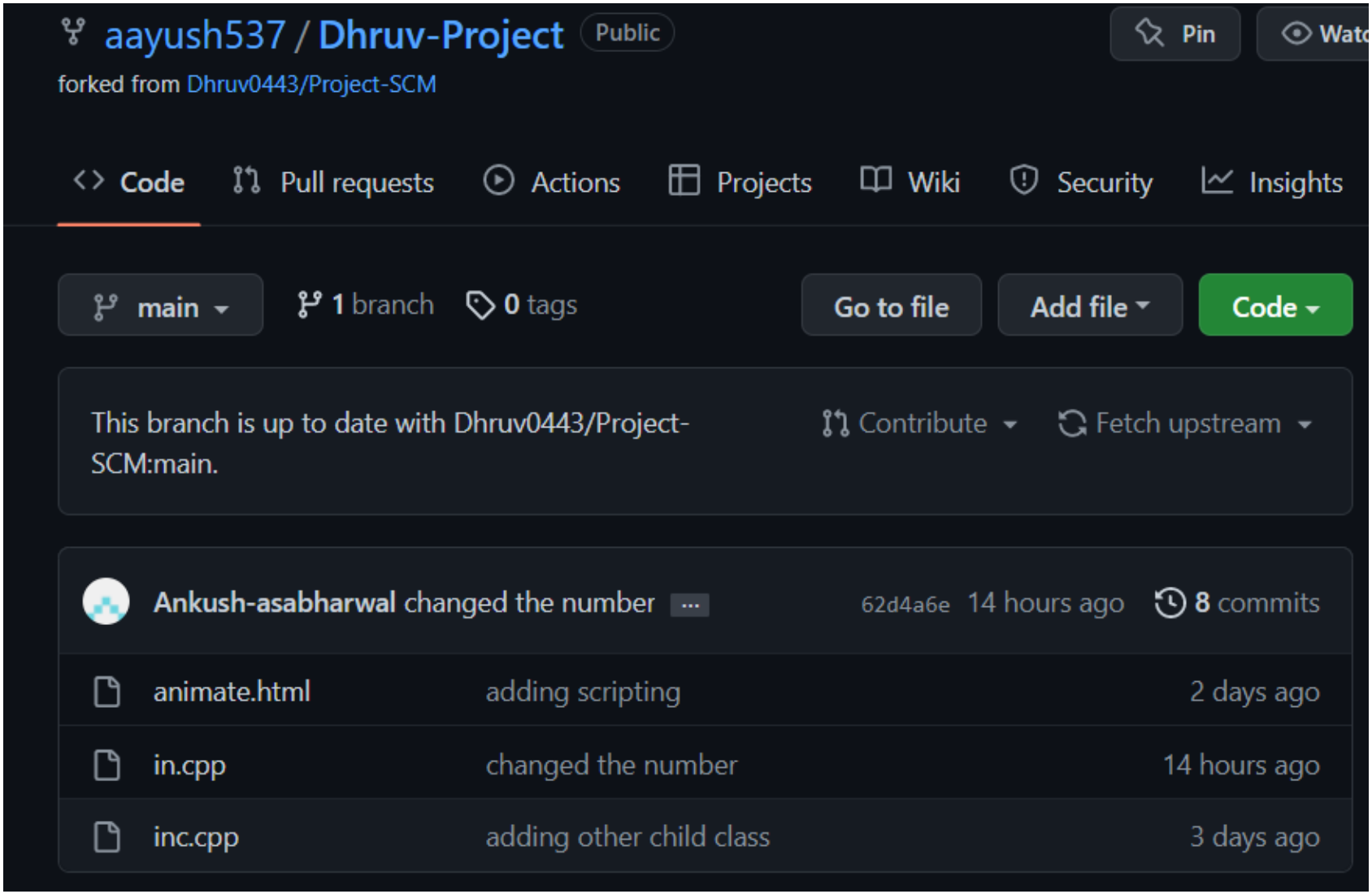


Step 2: Name the forked repository

```
HP@DESKTOP-D6NHS1S MINGW64 ~/Desktop/ForkedDhruv/Dhruv-Project (main)
$ git push -u origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 422 bytes | 422.00 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.
To https://github.com/aayush537/Dhruv-Project.git
   62d4a6e..de98055  main -> main
branch 'main' set up to track 'origin/main'.
```



Step3 : Now a repository is forked to your own account.



Step 4: Now initialize an empty repository.

```
HP@DESKTOP-D6NHS1S MINGW64 ~ (master)
$ cd C:/Users/HP/Desktop/Fork Dhruv
bash: cd: too many arguments

HP@DESKTOP-D6NHS1S MINGW64 ~ (master)
$ cd C:/Users/HP/Desktop/ForkedDhruv

HP@DESKTOP-D6NHS1S MINGW64 ~/Desktop/ForkedDhruv (master)
$ git init
Initialized empty Git repository in C:/Users/HP/Desktop/ForkedDhruv/.git/
```

Step 5: Clone the repository using git clone command.

```
HP@DESKTOP-D6NHS1S MINGW64 ~/Desktop/ForkedDhruv (master)
$ git clone https://github.com/aayush537/Dhruv-Project.git
Cloning into 'Dhruv-Project'...
remote: Enumerating objects: 25, done.
remote: Counting objects: 100% (25/25), done.
remote: Compressing objects: 100% (23/23), done.
remote: Total 25 (delta 6), reused 18 (delta 2), pack-reused 0
Receiving objects: 100% (25/25), done.
Resolving deltas: 100% (6/6), done.
```

Step 6: Now make some changes , push them to the staging area and commit them.

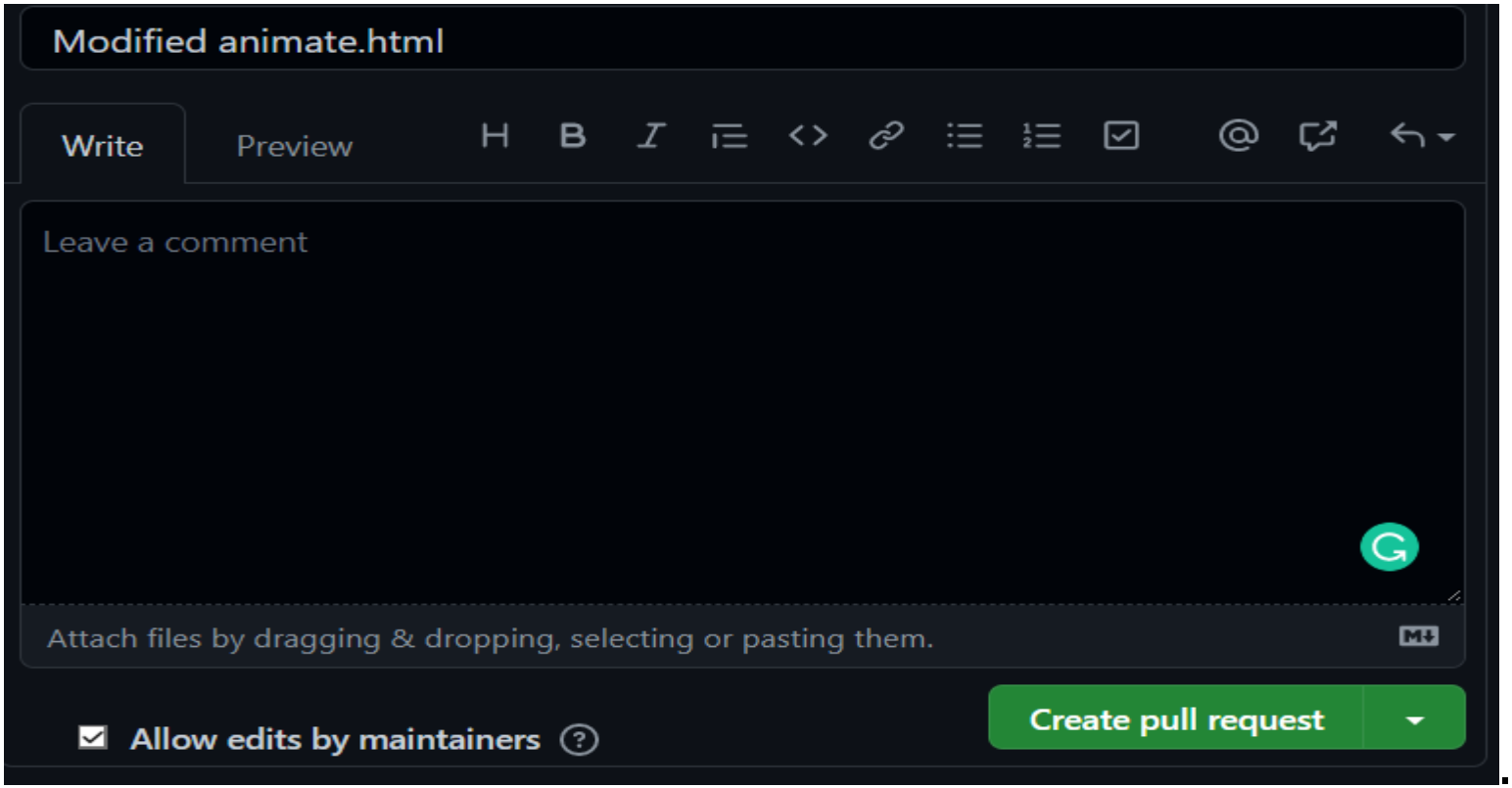
```
HP@DESKTOP-D6NHS1S MINGW64 ~/Desktop/ForkedDhruv (master)
$ code .

HP@DESKTOP-D6NHS1S MINGW64 ~/Desktop/ForkedDhruv (master)
$ git status
On branch master

No commits yet

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

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

HP@DESKTOP-D6NHS1S MINGW64 ~/Desktop/ForkedDhruv (master)
$ cd Dhruv-Project

HP@DESKTOP-D6NHS1S MINGW64 ~/Desktop/ForkedDhruv/Dhruv-Project (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

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:   animate.html

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

HP@DESKTOP-D6NHS1S MINGW64 ~/Desktop/ForkedDhruv/Dhruv-Project (main)
$ git commit -a -m "Modified animate.html"
[main de98055] Modified animate.html
 1 file changed, 9 insertions(+), 9 deletions(-)
```

Step7: Now push the file to your remote repository

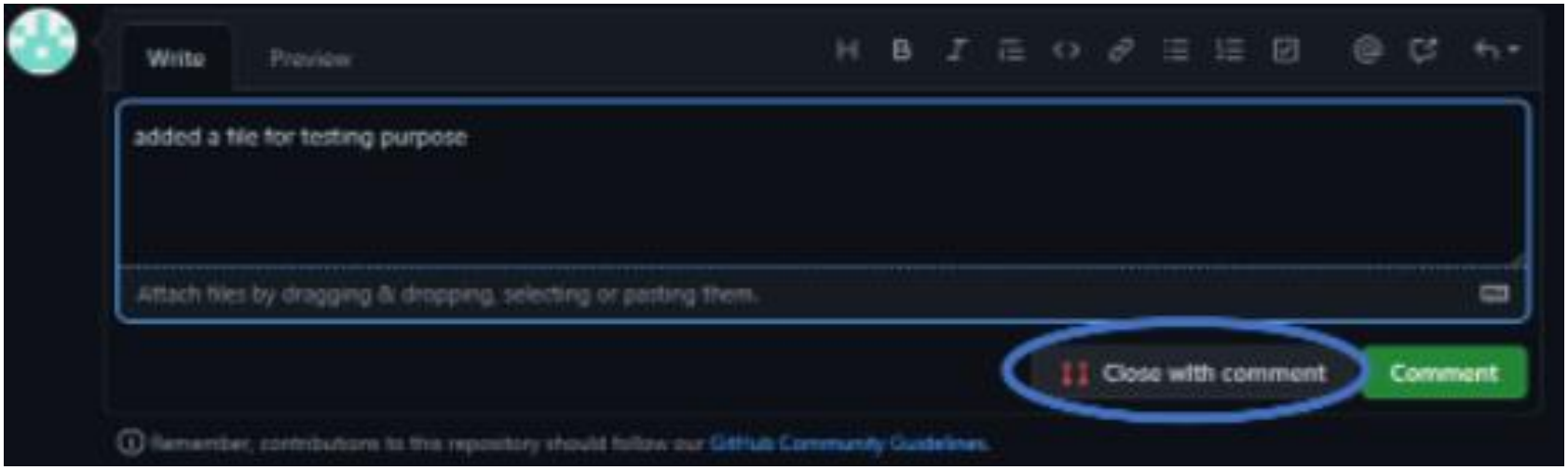
```
HP@DESKTOP-D6NHS1S MINGW64 ~/Desktop/ForkedDhruv/Dhruv-Project (main)
$ git push -u origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 422 bytes | 422.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/aayush537/Dhruv-Project.git
   62d4a6e..de98055  main -> main
branch 'main' set up to track 'origin/main'.
```

Step 8: After pushing the file on Github will either automatically ask you to create a pull request or you can create your own pull request.

Step9: Github will detect any conflicts and ask you to enter a description of your pull request



Step 10: If we chooses not to merge the pull request we will close the pull request. To close the pull request we simply click on close pull request and add comment/reason why we closed the pull request.



Aim: Create a Pull Request on a Team Members Repository and close Pull Requests generated by Team Members on your own Repository as a Maintainer

Creating a pull request on Dhruv0443/Project-SCM

- First we have to fork his repository

Dhruv0443 / Project-SCM Public

Watch 1 | Fork 0 | Star 0

Code | Issues | Pull requests | Actions | Projects | Wiki | Security | Insights

main | 1 branch | 0 tags | Go to file | Add file | Code

Ankush-asabharwal changed the number ... 62d4a6e 14 hours ago 8 commits

animate.html	adding scripting	2 days ago
in.cpp	changed the number	14 hours ago
inc.cpp	adding other child class	3 days ago

Help people interested in this repository understand your project by adding a README. Add a README

About

No description, website, or topics provided.

0 stars

1 watching

0 forks

Releases

No releases published

Create a new release

Dhruv0443 / Project-SCM 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 * aayush537 / **Repository name** * Dhruv Project

Your new repository will be created as Dhruv-Project.

By default, forks are named tomize the name to distinguish it further.

Description (optional)

Activate Windows

- After Forking,create a new folder

- Now,Put the address of this folder in your bash.
- Then Initialize a new repo using “git init”

```
HP@DESKTOP-D6NHS1S MINGW64 ~ (master)
$ cd C:/Users/HP/Desktop/Fork Dhruv
bash: cd: too many arguments

HP@DESKTOP-D6NHS1S MINGW64 ~ (master)
$ cd C:/Users/HP/Desktop/ForkedDhruv

HP@DESKTOP-D6NHS1S MINGW64 ~/Desktop/ForkedDhruv (master)
$ git init
Initialized empty Git repository in C:/Users/HP/Desktop/ForkedDhruv/.git/
```

Step 2: Clone his repo on local space

```
HP@DESKTOP-D6NHS1S MINGW64 ~/Desktop/ForkedDhruv (master)
$ git clone https://github.com/aayush537/Dhruv-Project.git
Cloning into 'Dhruv-Project'...
remote: Enumerating objects: 25, done.
remote: Counting objects: 100% (25/25), done.
remote: Compressing objects: 100% (23/23), done.
remote: Total 25 (delta 6), reused 18 (delta 2), pack-reused 0
Receiving objects: 100% (25/25), done.
Resolving deltas: 100% (6/6), done.
```

Step 3 : Make some changes and commit

```
HP@DESKTOP-D6NHS1S MINGW64 ~/Desktop/ForkedDhruv (master)
$ code .

HP@DESKTOP-D6NHS1S MINGW64 ~/Desktop/ForkedDhruv (master)
$ git status
On branch master

No commits yet

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

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

HP@DESKTOP-D6NHS1S MINGW64 ~/Desktop/ForkedDhruv (master)
$ cd Dhruv-Project

HP@DESKTOP-D6NHS1S MINGW64 ~/Desktop/ForkedDhruv/Dhruv-Project (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

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:   animate.html

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

HP@DESKTOP-D6NHS1S MINGW64 ~/Desktop/ForkedDhruv/Dhruv-Project (main)
$ git commit -a -m "Modified animate.html"
[main de98055] Modified animate.html
 1 file changed, 9 insertions(+), 9 deletions(-)
```

Step 4 : Push the changes to the forked repo

```
HP@DESKTOP-D6NHS1S MINGW64 ~/Desktop/ForkedDhruv/Dhruv-Project (main)
$ git push -u origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 422 bytes | 422.00 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.
To https://github.com/aayush537/Dhruv-Project.git
   62d4a6e..de98055  main -> main
branch 'main' set up to track 'origin/main'.
```

Step 5: Make the pull request

Modified animate.html

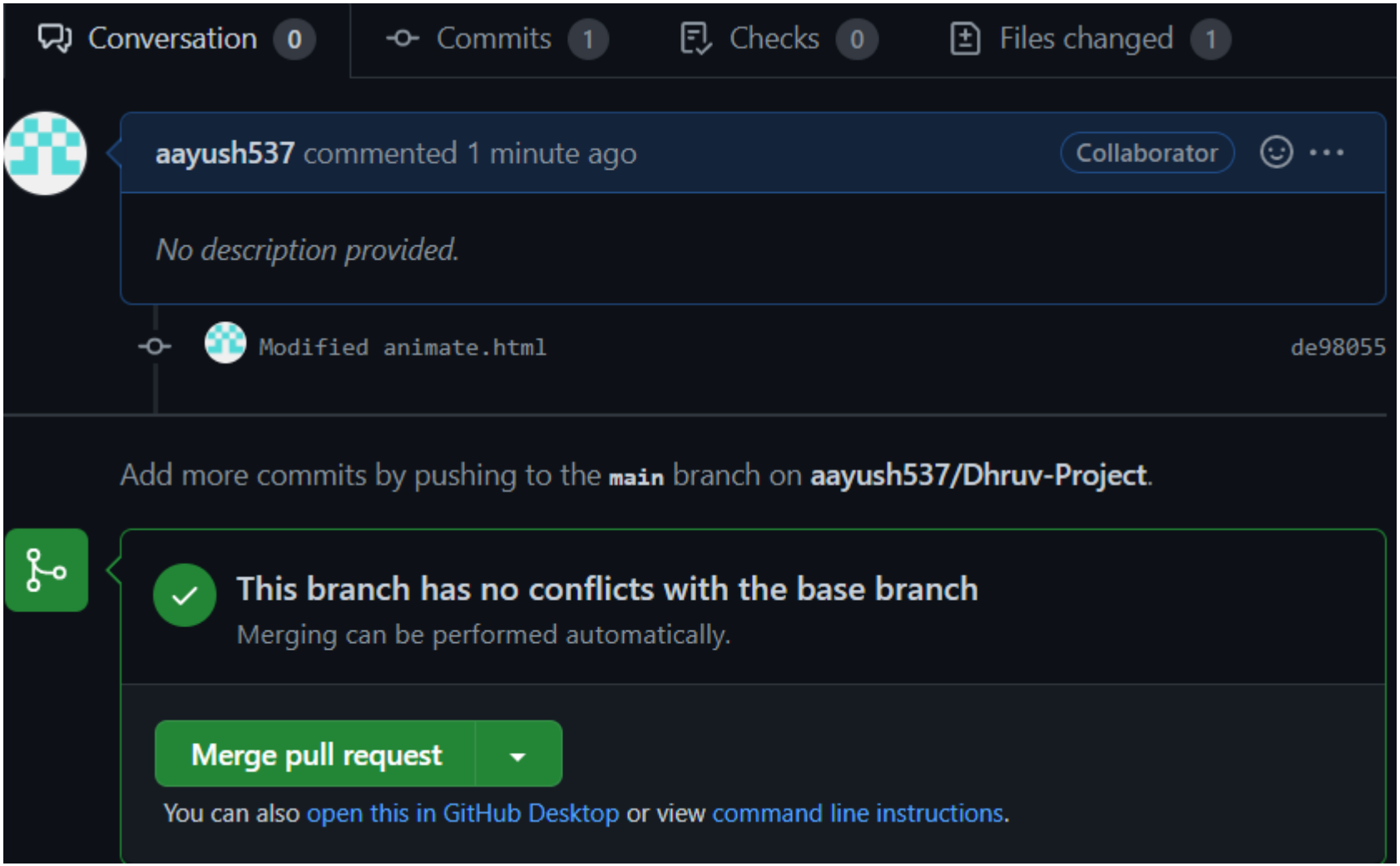
Write Preview

Leave a comment

Attach files by dragging & dropping, selecting or pasting them.

Allow edits by maintainers

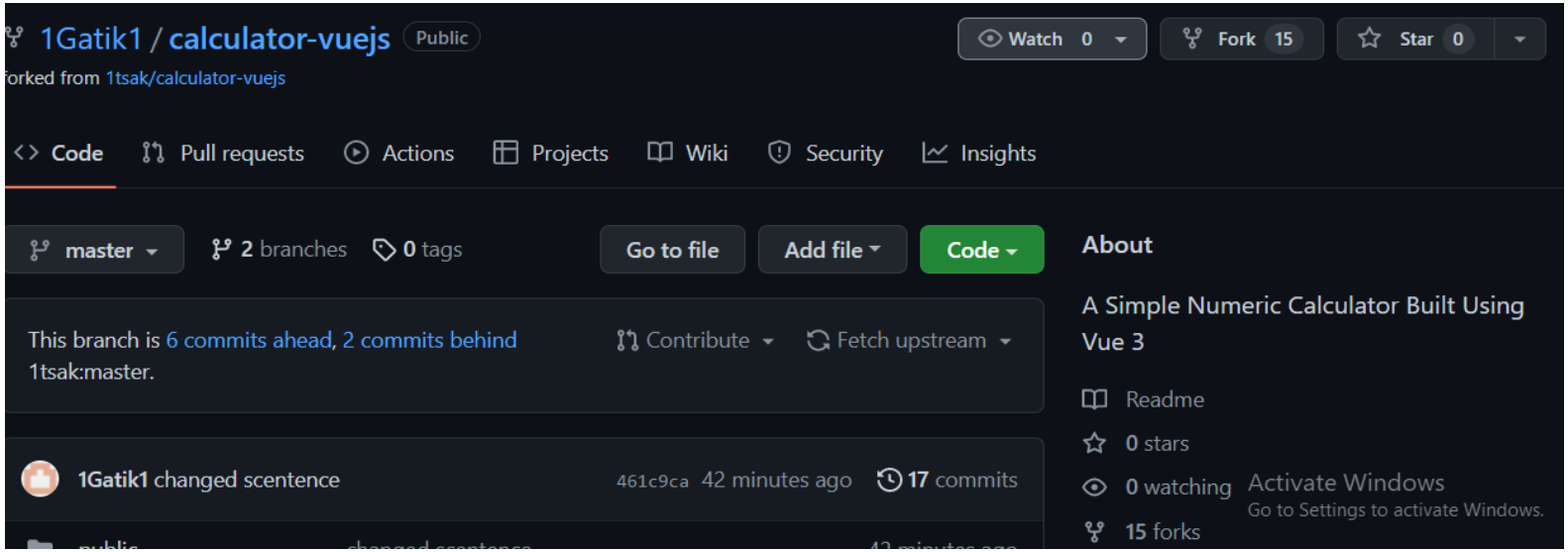
Create pull request

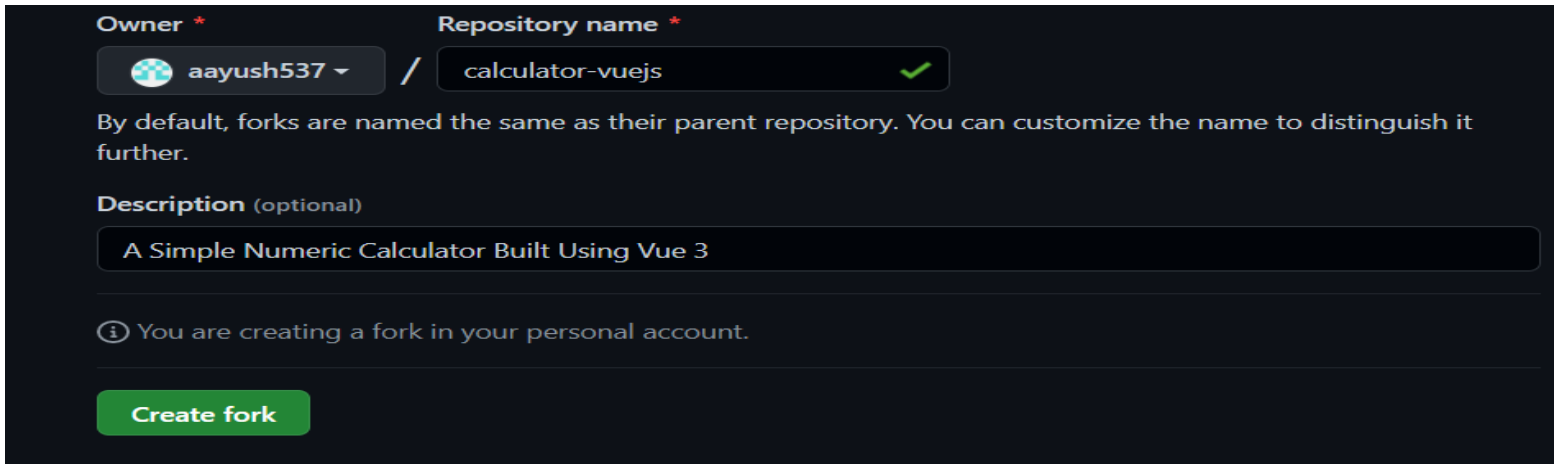


After creating a pull request the owner of the repo recieves an email whether he want to merge the pull request or close it.

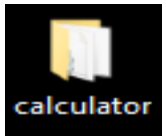
Creating a pull request on

- First we have to fork his repository





- After Forking,create a new folder



- Now,Put the address of this folder in your bash.
- Then Initialize a new repo using “git init”

Step 2: Clone his repo on local space

```
$ git clone https://github.com/aayush537/calculator-vuejs.git
Cloning into 'calculator-vuejs'...
remote: Enumerating objects: 80, done.
remote: Counting objects: 100% (80/80), done.
remote: Compressing objects: 100% (60/60), done.
remote: Total 80 (delta 32), reused 63 (delta 19), pack-reused 0
Receiving objects: 100% (80/80), 233.40 KiB | 1.34 MiB/s, done.
Resolving deltas: 100% (32/32), done.
```

Step 3 : Make some changes and commit

```
$ cd calculator-vuejs

HP@DESKTOP-D6NHS1S MINGW64 ~/Desktop/Vue/calculator-vuejs (master)
$ ls
README.md  babel.config.js  jsconfig.json  package-lock.json  package.json  public/  src/  vue.config.js

HP@DESKTOP-D6NHS1S MINGW64 ~/Desktop/Vue/calculator-vuejs (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:   public/index.html
```

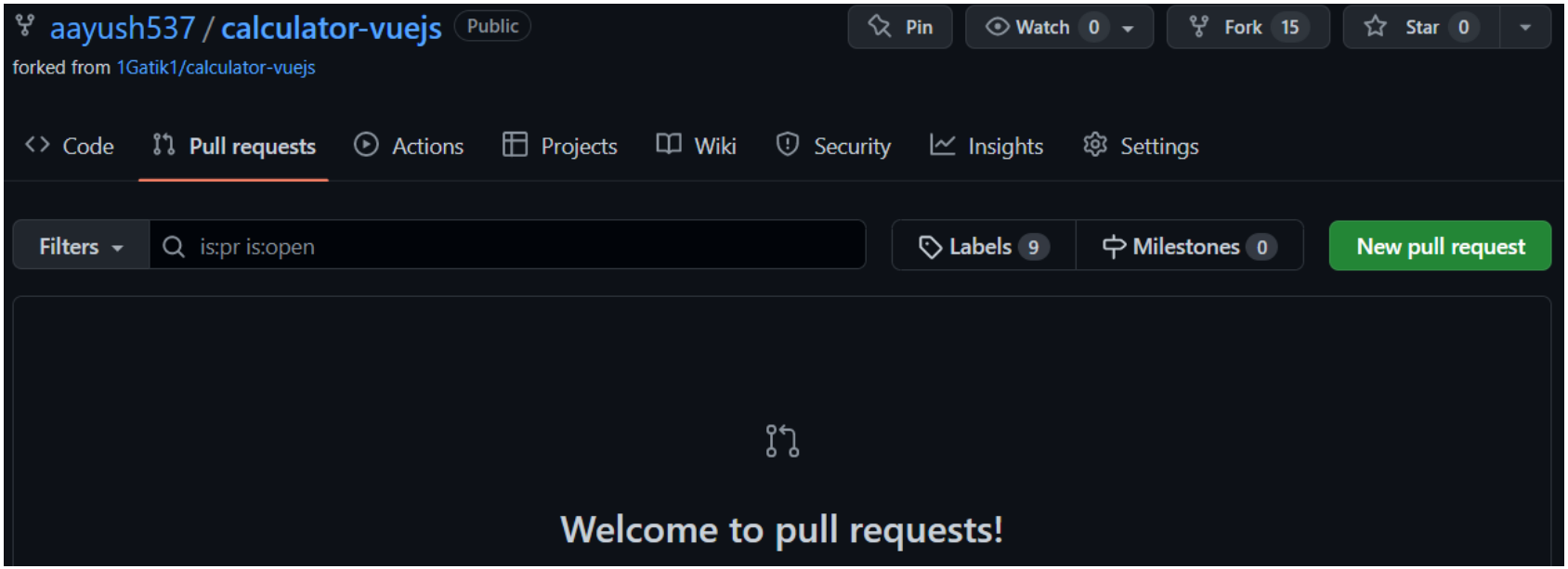
```
HP@DESKTOP-D6NHS1S MINGW64 ~/Desktop/Vue/calculator-vuejs (features)
$ git status
On branch features
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   README.md
        modified:   vue.config.js

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

HP@DESKTOP-D6NHS1S MINGW64 ~/Desktop/Vue/calculator-vuejs (features)
$ git add -A

HP@DESKTOP-D6NHS1S MINGW64 ~/Desktop/Vue/calculator-vuejs (features)
$ git status
On branch features
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   README.md
        modified:   vue.config.js
```

```
$ git add -A

HP@DESKTOP-D6NHS1S MINGW64 ~/Desktop/Vue/calculator-vuejs (Gatik)
$ git status
On branch Gatik
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   public/index.html
        modified:   src/assets/youtube-light.svg


HP@DESKTOP-D6NHS1S MINGW64 ~/Desktop/Vue/calculator-vuejs (Gatik)
$ git commit -m "Editing gatik's file"
[Gatik 346218b] Editing gatik's file
 2 files changed, 2 insertions(+), 2 deletions(-)
```

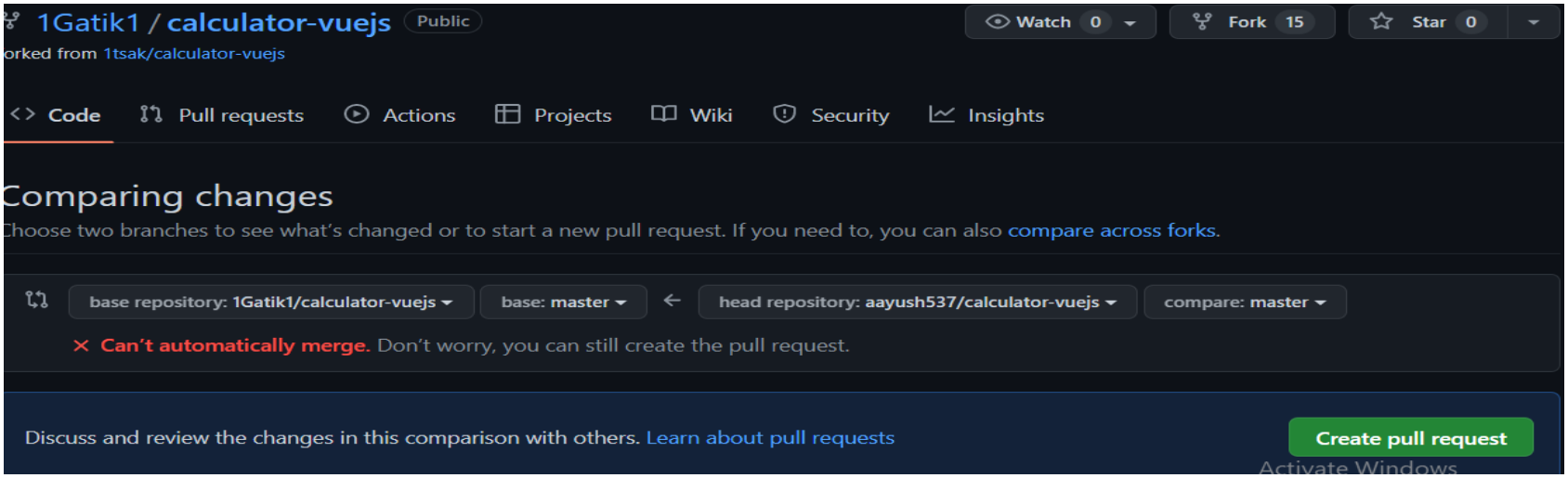
Step 4 : Push the changes to the forked repo

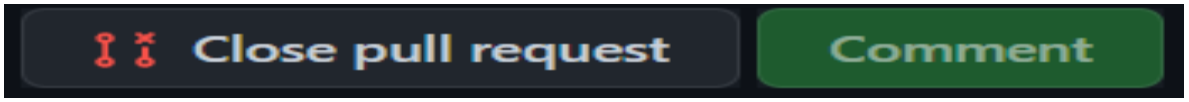
```
$ git push -u origin Gatik
Enumerating objects: 27, done.
Counting objects: 100% (27/27), done.
Delta compression using up to 4 threads
Compressing objects: 100% (19/19), done.
Writing objects: 100% (19/19), 1.83 KiB | 469.00 KiB/s, done.
Total 19 (delta 11), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (11/11), completed with 6 local objects.
remote:
remote: Create a pull request for 'Gatik' on GitHub by visiting:
remote:      https://github.com/aayush537/calculator-vuejs/pull/new/Gatik
remote:
To https://github.com/aayush537/calculator-vuejs.git
 * [new branch]      Gatik -> Gatik
branch 'Gatik' set up to track 'origin/Gatik'.
```

Step 5: Click on new pull request


aayush537 / calculator-vuejs
Public
forked from 1Gatik1/calculator-vuejs
Pin
Watch 0
Fork 15
Star 0
Code
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
Filters
is:pr is:open
Labels 9
Milestones 0
New pull request
Welcome to pull requests!

Step 6


1Gatik1 / calculator-vuejs
Public
orked from 1tsak/calculator-vuejs
Watch 0
Fork 15
Star 0
Code
Pull requests
Actions
Projects
Wiki
Security
Insights
Comparing changes
Choose two branches to see what's changed or to start a new pull request. If you need to, you can also compare across forks.
base repository: 1Gatik1/calculator-vuejs
base: master
head repository: aayush537/calculator-vuejs
compare: master
Can't automatically merge. Don't worry, you can still create the pull request.
Discuss and review the changes in this comparison with others. Learn about pull requests
Create pull request



After creating a pull request the owner of the repo recieves an email whether he want to merge that committed changes or not.

Closing the pull request generated by Ankusha

My team members forked my repository and made some changes in that and a pull request is generated As a result I received an email asking whether I want to have those changes in my own main repo or not .

➔Ankush-asabharwal made this pull request :

Ankush-asabharwal <notifications@github.com> Unsubscribe
to aayush537/SCM-Project, Subscribed

You can view, comment on, or merge this pull request online at:

https://github.com/aayush537/SCM-Project/pull/3

Commit Summary

- 998bc18 Update quicksort.cpp

File Changes (1 file)

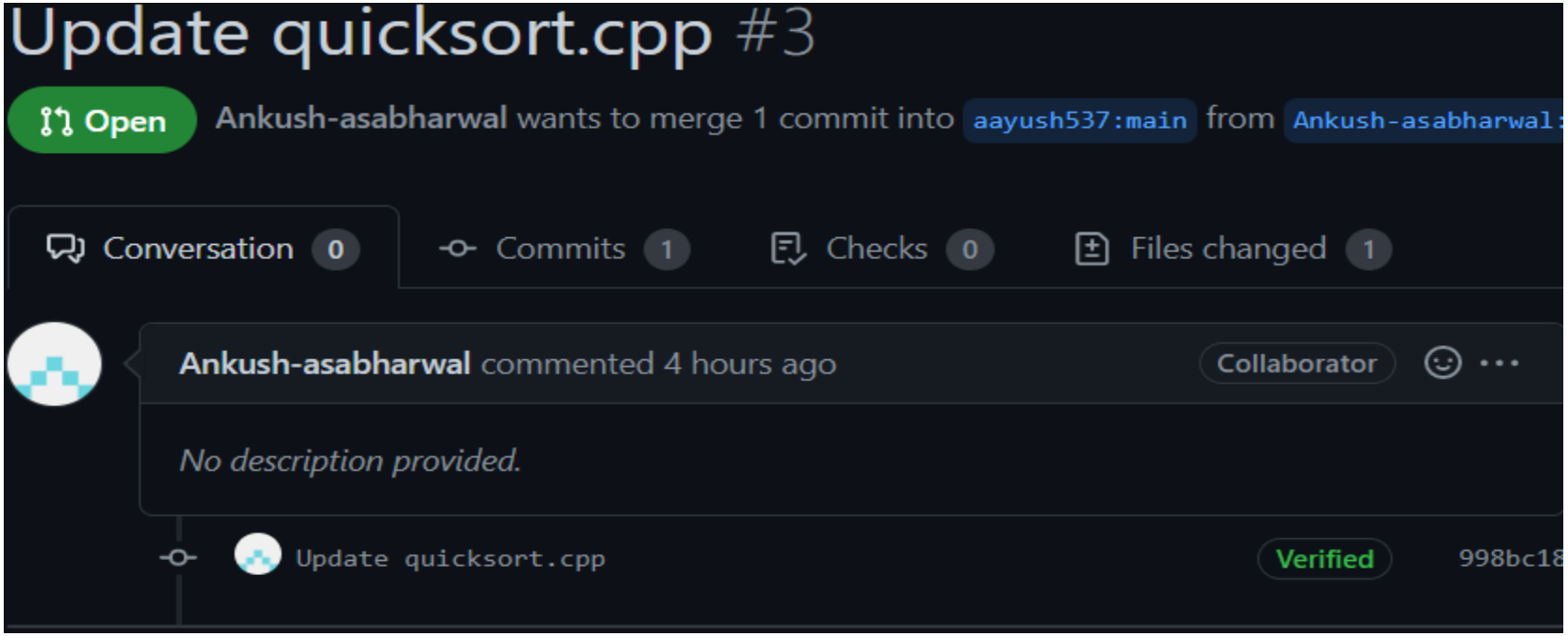
- **M** quicksort.cpp (1)

Patch Links:

- https://github.com/aayush537/SCM-Project/pull/3.patch
- https://github.com/aayush537/SCM-Project/pull/3.diff

—
Reply to this email directly, view it on GitHub, or unsubscribe.
You are receiving this because you are subscribed to this thread.

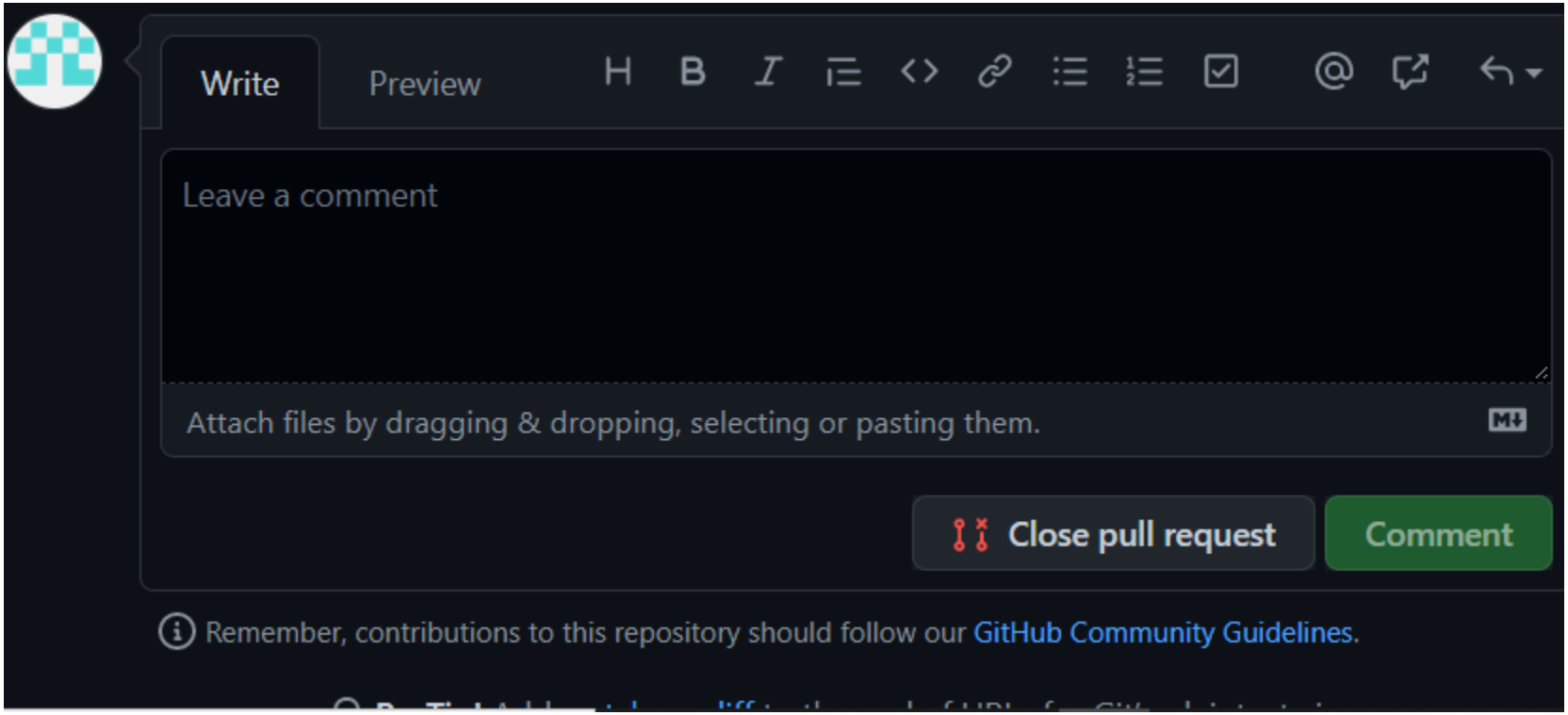
➔ After visiting the link you'll be Able to see the change and merge option



➔ If you want to merge that change, you can merge

➔ If you want to close that pull request made by your team member without merging you can close as well .

➔ So as a maintainer of the repo I am closing the pull request made by my team member



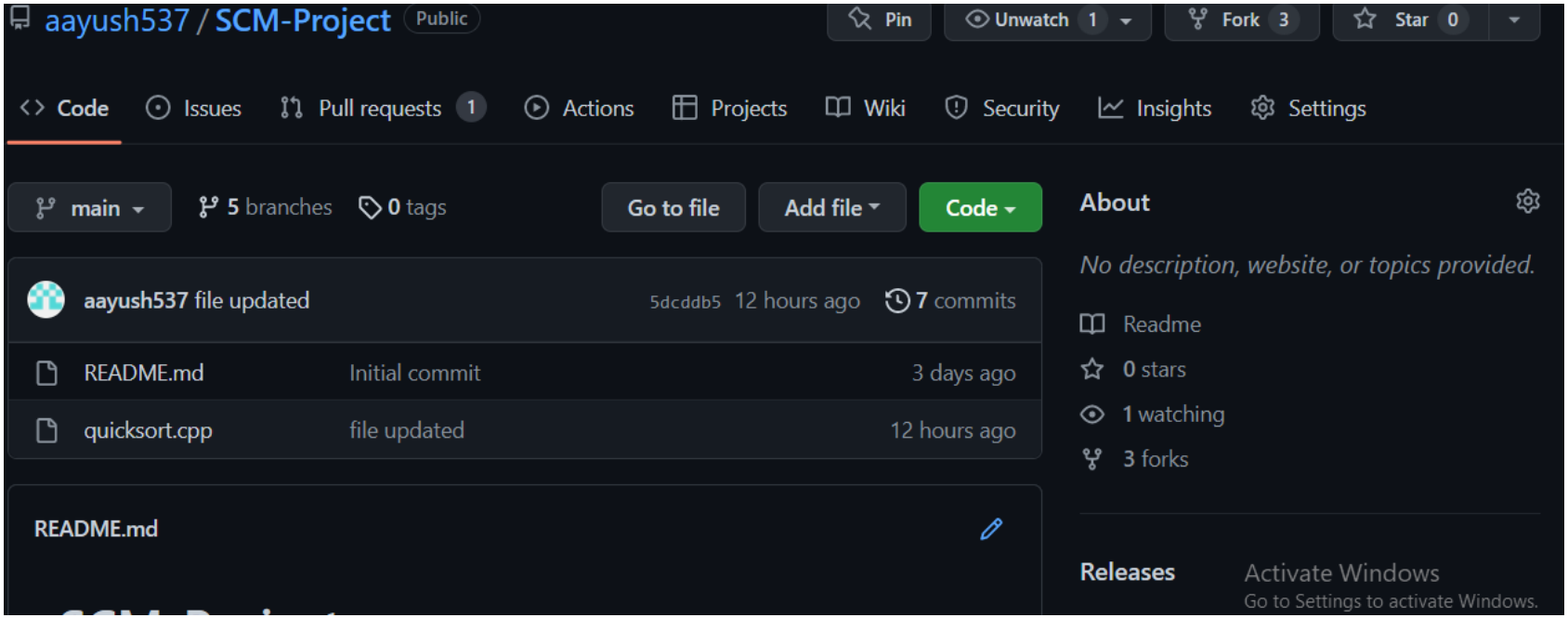
➔ If you want to reopen that pull request you can reopen.

Aim: Publish and Print Network Graphs

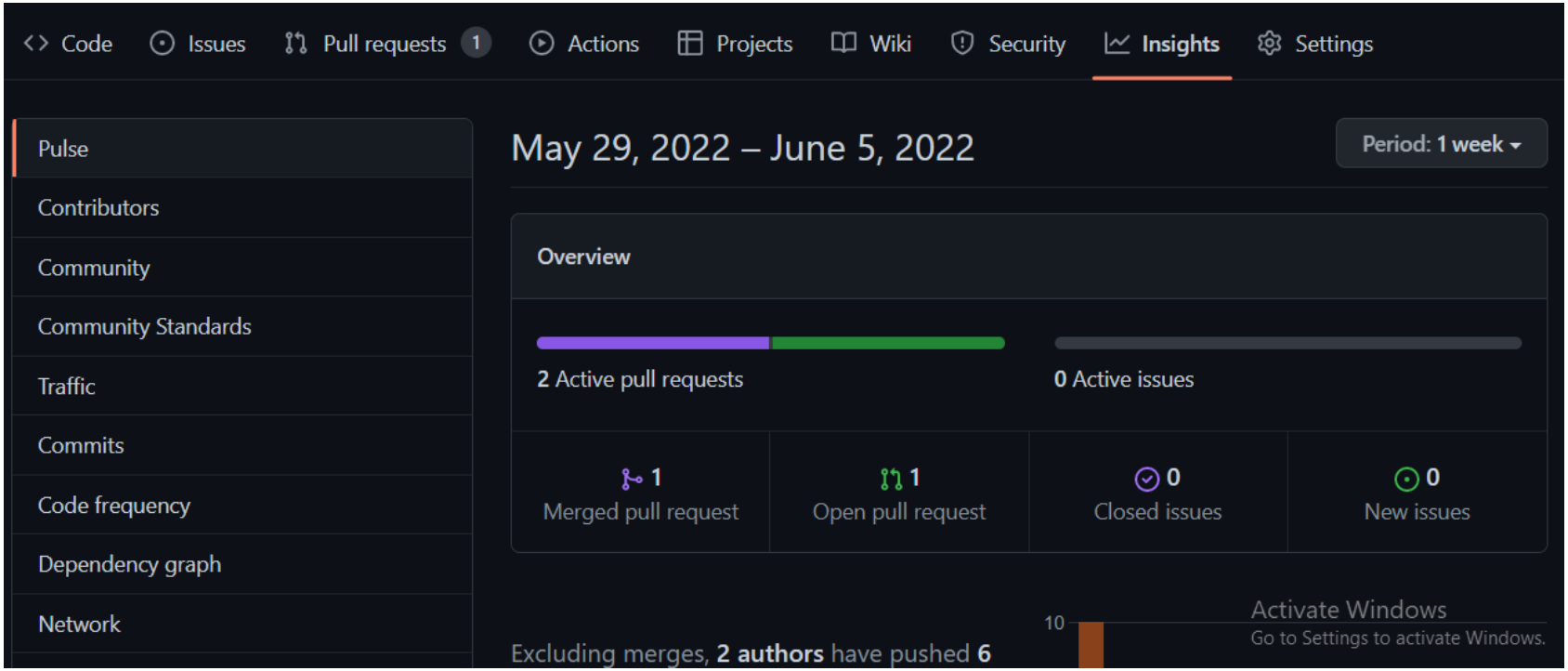
The network graph is one of the useful features for developers on Github. It is used to display the branch history of the entire repository network ,including branches of the root repository and branches of forks that are unique to the network

Accessing the network graph :

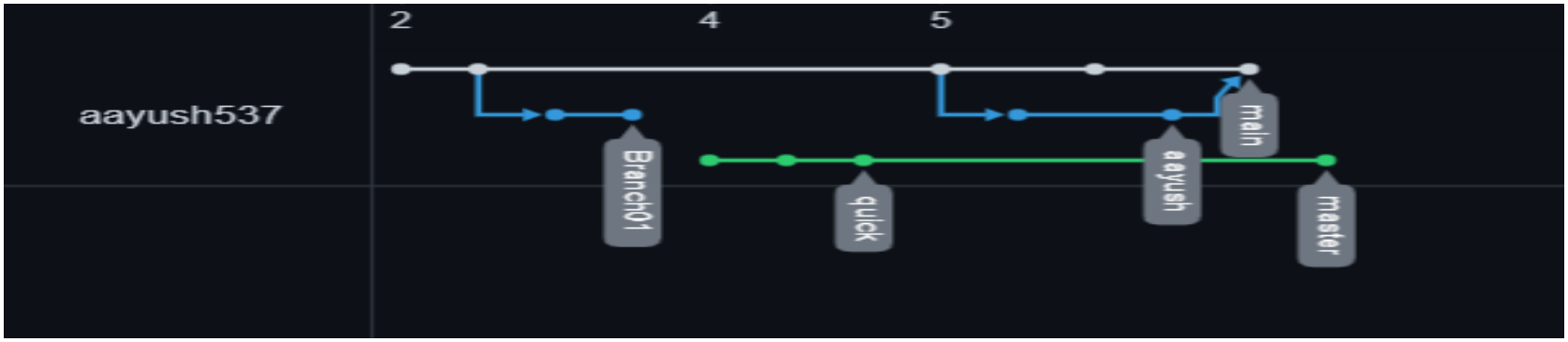
➢On Github.com, navigate to the main page of the repository .



➢Under your repository name, click insights



➢In the left sidebar , click Networks



You will get the network graph of your repository which displays the branch history of the entire repository network, including branches of the root repository and branches of forks that contain commits unique to the network..