



# **A Project report**

on

**“Project from task1.1,1.2,2.0”**

with

**Source Code Management**

**(CS181)**

Submitted by

**ARMAAN SINGH BHASIN**

**2110990258**

**G8-A**





## **Department of Computer Science & Engineering**

Chitkara University Institute of Engineering and Technology, Punjab

Jan- June  
(2021-22)

Institute/School Name	<b>Chitkara University Institute of Engineering and Technology</b>		
Department Name	<b>Department of Computer Science &amp; Engineering</b>		
Programme Name	<b>Bachelor of Engineering (B.E.), Computer Science &amp; Engineering</b>		
Course Name	<b>Source Code Management</b>	Session	<b>2021-22</b>
Course Code	<b>CS181</b>	Semester/Batch	<b>2<sup>nd</sup>/2021</b>
Submitted to	<b>Dr. Monit kapoor</b>	Group No	<b>G8 -A</b>

Name: ARMAAN SINGH BHASIN  
ROLL NO - 2110990258



## Table of Content-

<b>S. No.</b>	<b>Title</b>	<b>Page No.</b>
1	Version control with Git	4
1.1	Installation of GIT	4 - 11
1.2	Setting Up GitHub Account	12 - 13
1.3	Configuration of Git	14 - 15
2.	Program	15 - 30
2.1	Making GIT Repository	16 - 20
2.2	Create and Visualize Branches	20 - 23
2.3	Git Lifecycle Description	23 - 25
2.4	UPLOADING DATA ON GITHUB	25 - 30
2.5	Add collaborators on GitHub Repo	31 -33
2.6	Fork and Commit	33 - 35
2.7	Merge and Resolve conflicts created due to own activity and collaborators activity	36 -44
2.8	Reset	45 - 47
2.9	Git ignore	47 - 48
3.0	Create a distributed Repository and add members in project team	48-53
3.1	Open and close a pull request.	54 -56



3.2	Atleast One of project member shall create a pull request on a team members repo and close pull requests generated by team members on own Repo	56-57
3.3	Publish and print network graph	57-58



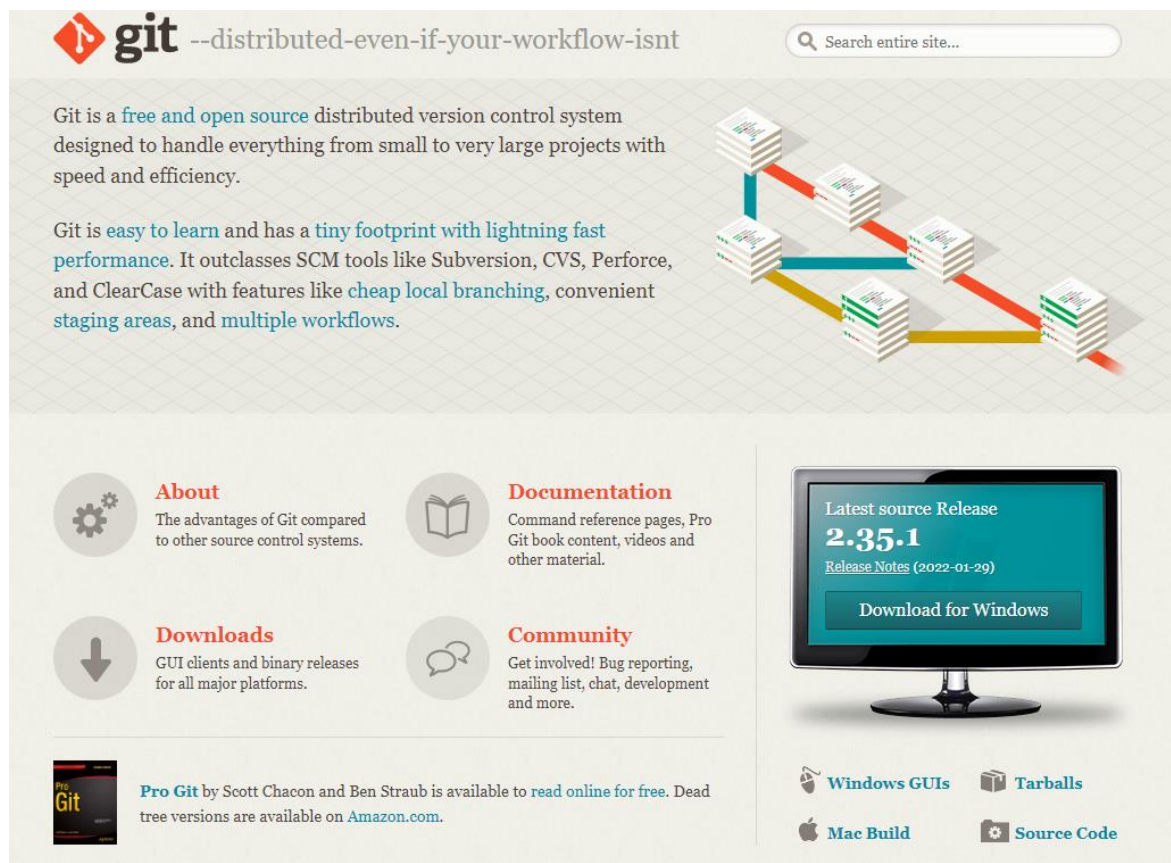
## 1.1 Installation of GIT

### Step 1)

To download the Git installer, visit the Git official site and go to the download page.

The link for the download page is <https://git-scm.com/downloads>

The page looks like as: -



Click on the package given on the page as **download 2.35.1 for windows**. The download will start after selecting the package.

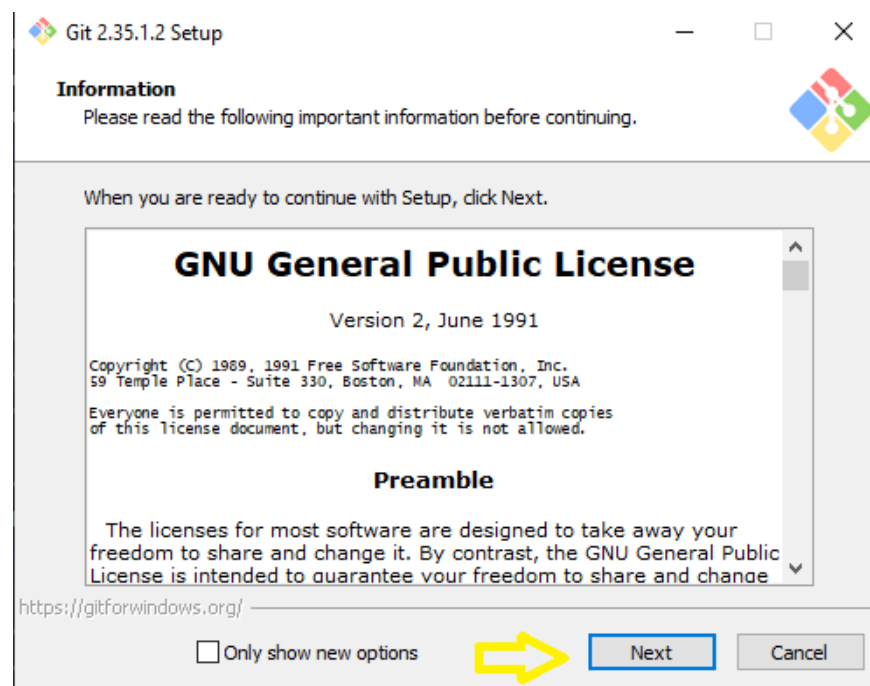
Now, the Git installer package has been downloaded.



## Step 2)

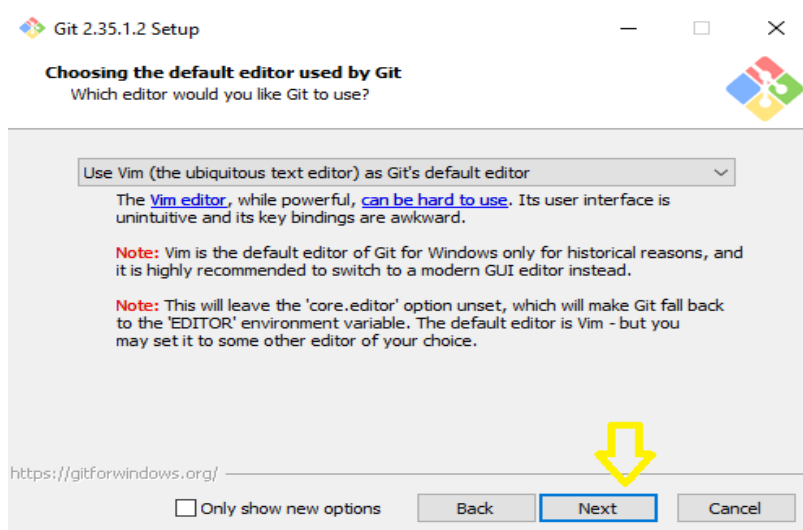
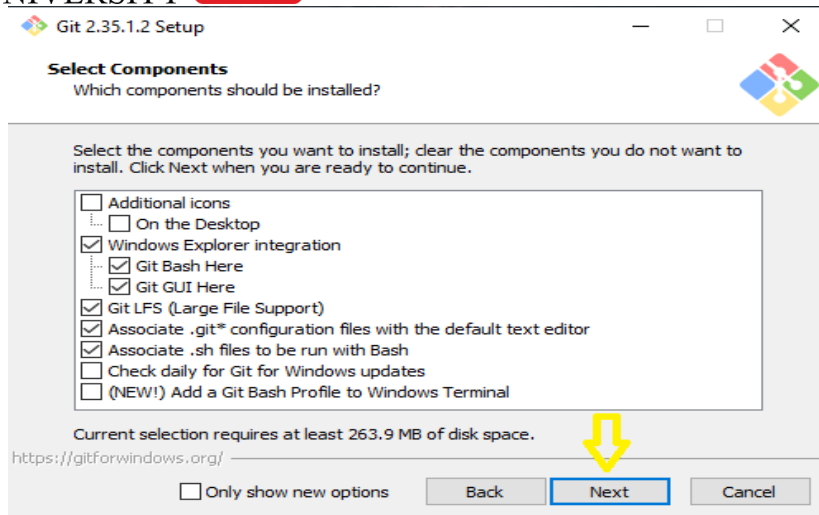
Click on the download installer file and then Provide Admin Permission click on next.

The page looks like as: -



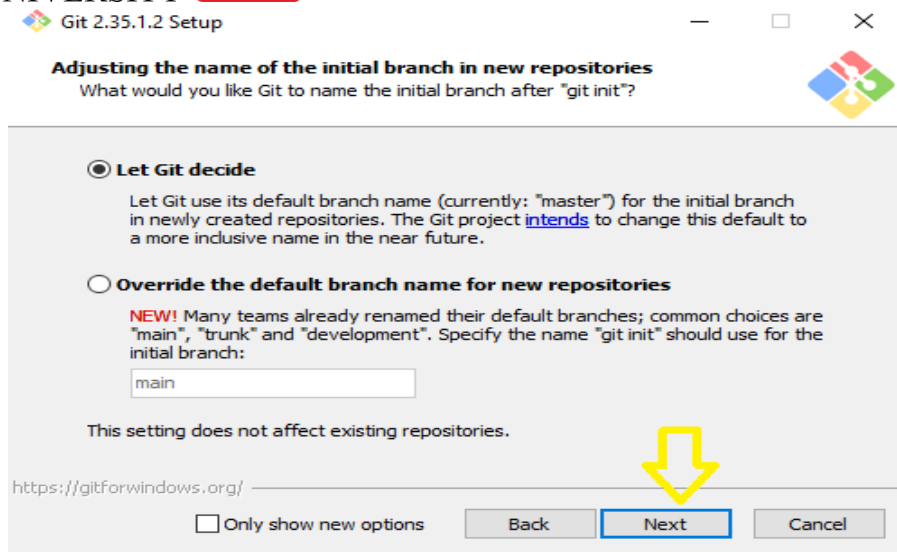
## Step 3)

Simply click on the next button as it automatically selects the required file. The page looks like as:



## Step 4)

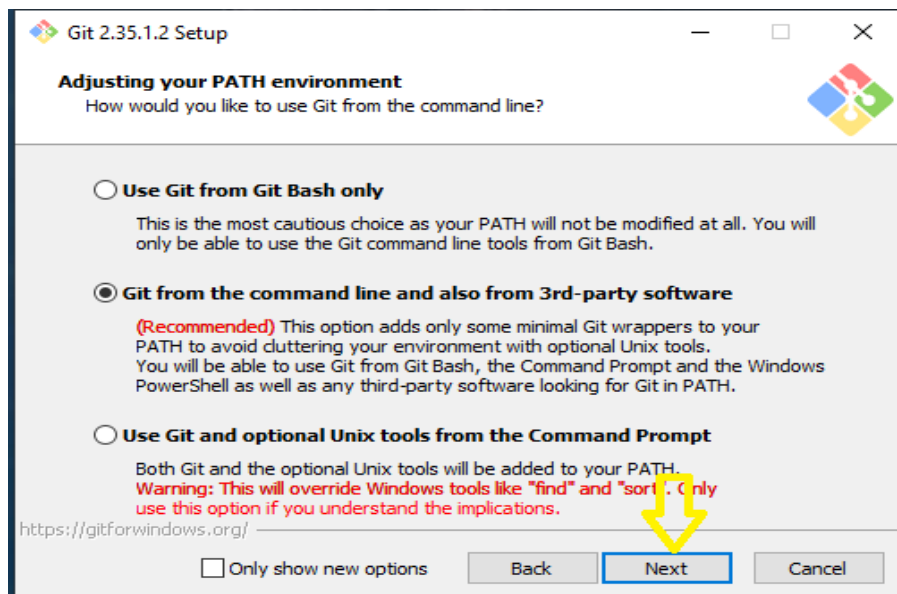
You can choose your preferred choice. Click next to continue.  
The page looks like as: -



## Step 5)

**Note:** - Just simply click on next as it automatically selects the required file.

The page looks like as: -

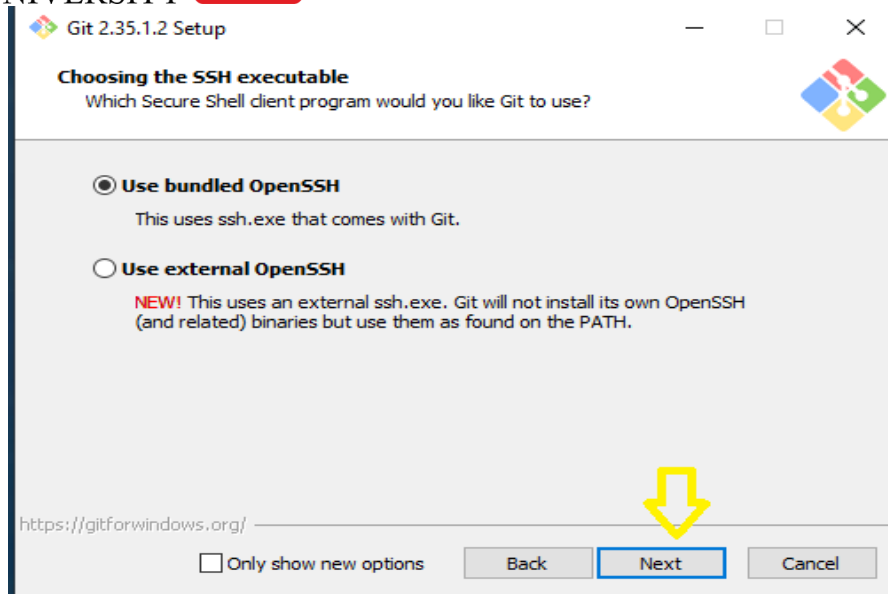


## Step6)

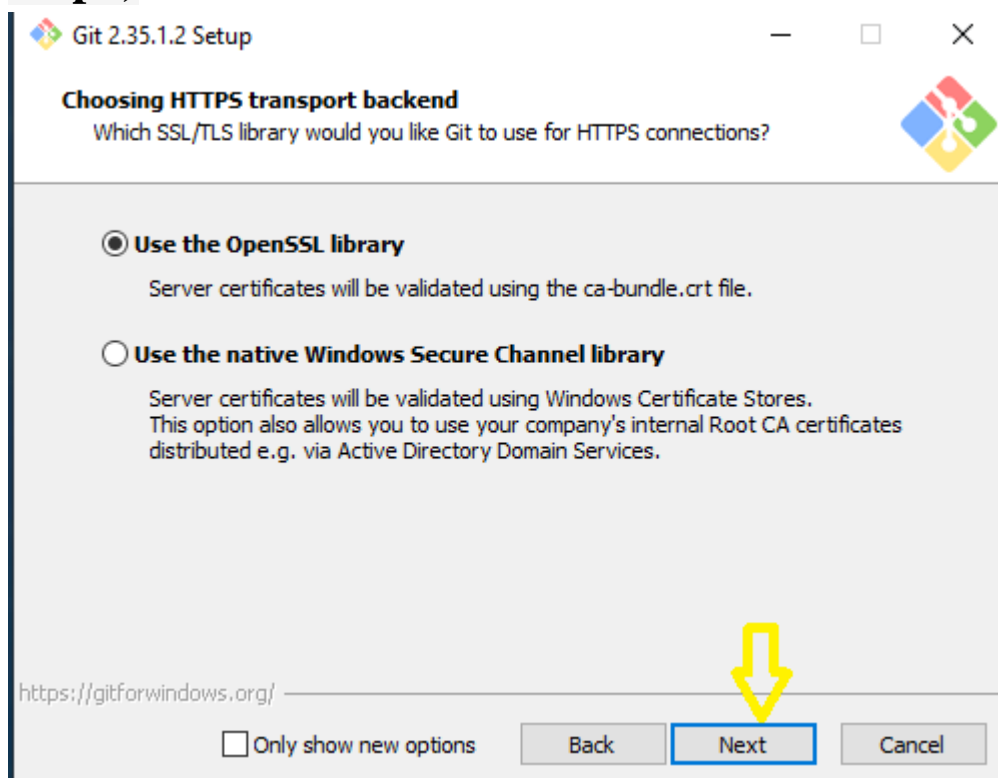
The Git is getting download in your system

The page looks like as: -





## Step7)



## Step8)

### Configuring the line ending conversions

How should Git treat line endings in text files?

☒ **Checkout Windows-style, commit Unix-style line endings**

Git will convert LF to CRLF when checking out text files. When committing text files, CRLF will be converted to LF. For cross-platform projects, this is the recommended setting on Windows ("core.autocrlf" is set to "true").

☐ **Checkout as-is, commit Unix-style line endings**

Git will not perform any conversion when checking out text files. When committing text files, CRLF will be converted to LF. For cross-platform projects, this is the recommended setting on Unix ("core.autocrlf" is set to "input").

☐ **Checkout as-is, commit as-is**

Git will not perform any conversions when checking out or committing text files. Choosing this option is not recommended for cross-platform projects ("core.autocrlf" is set to "false").

<https://gitforwindows.org/>

☐ Only show new options

Back

Next

Cancel

## Step9)

### Choose the default behavior of `git pull`

What should `git pull` do by default?

☒ **Default (fast-forward or merge)**

This is the standard behavior of `git pull`: fast-forward the current branch to the fetched branch when possible, otherwise create a merge commit.

☐ **Rebase**

Rebase the current branch onto the fetched branch. If there are no local commits to rebase, this is equivalent to a fast-forward.

☐ **Only ever fast-forward**

Fast-forward to the fetched branch. Fail if that is not possible.

<https://gitforwindows.org/>

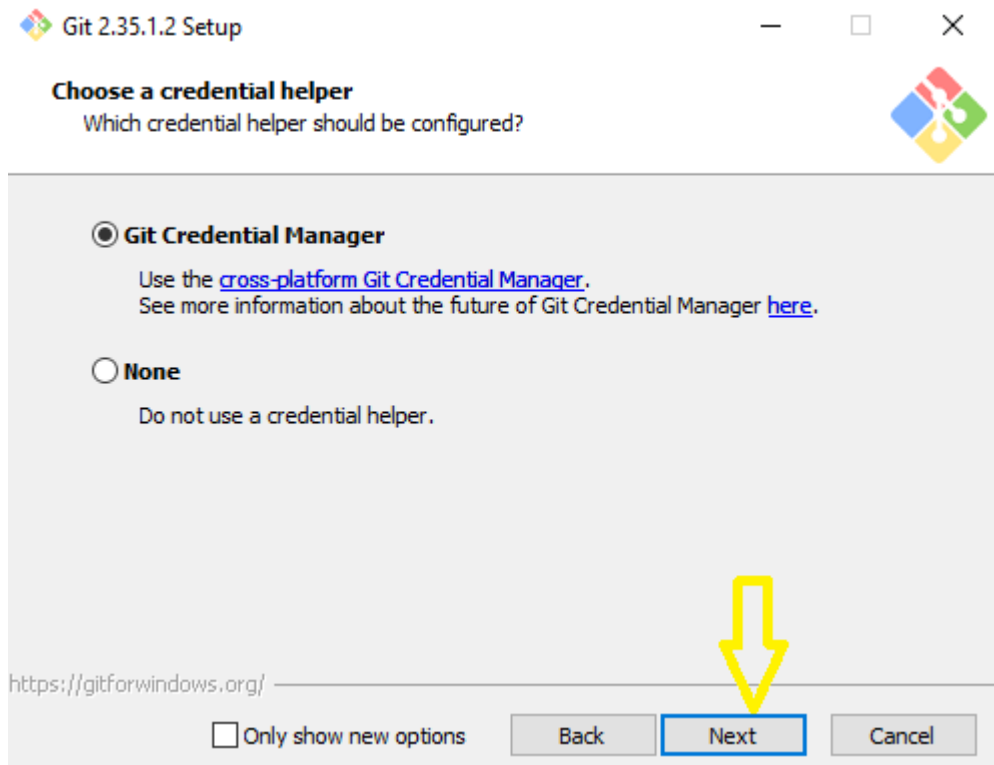
☐ Only show new options

Back

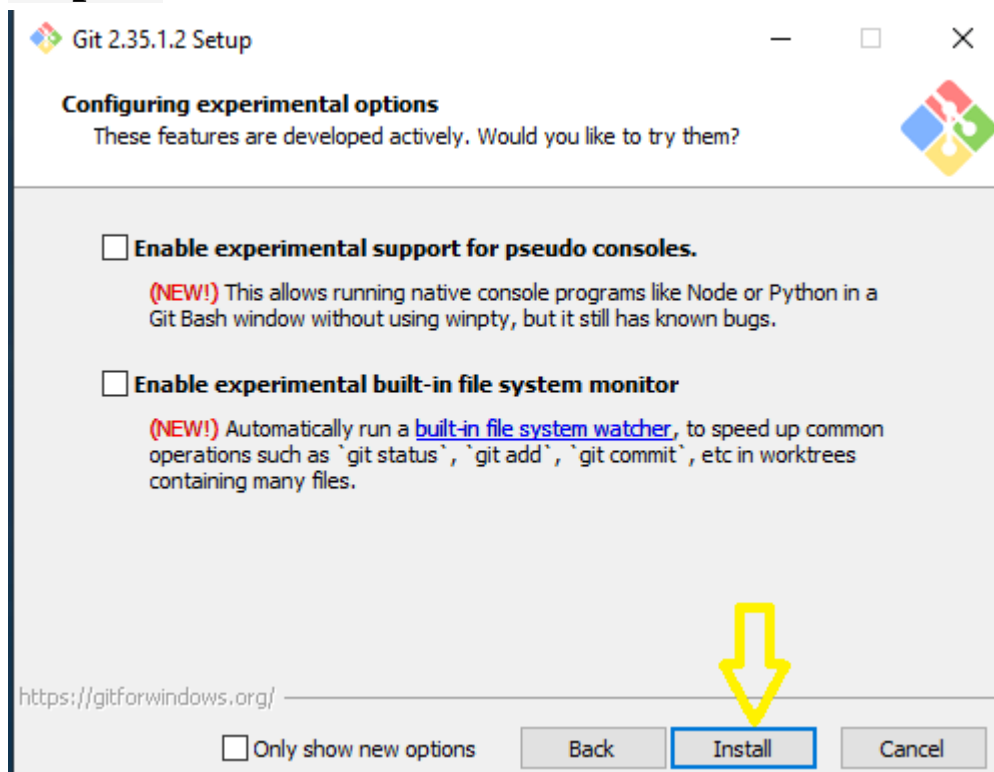
Next

Cancel

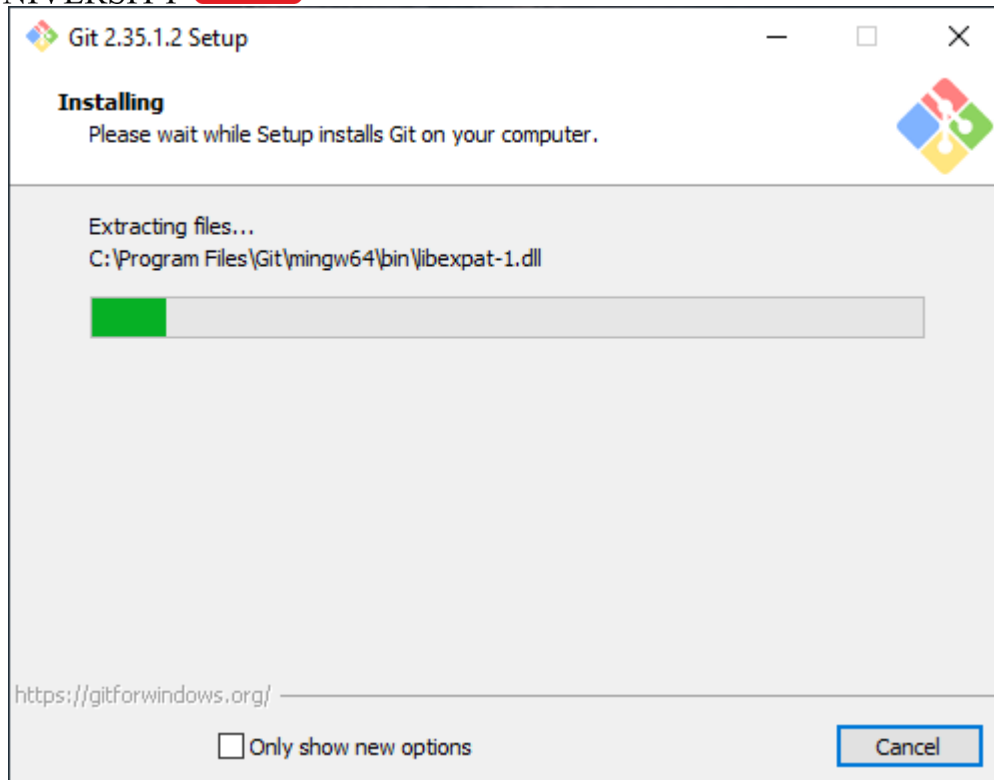
## Step10)



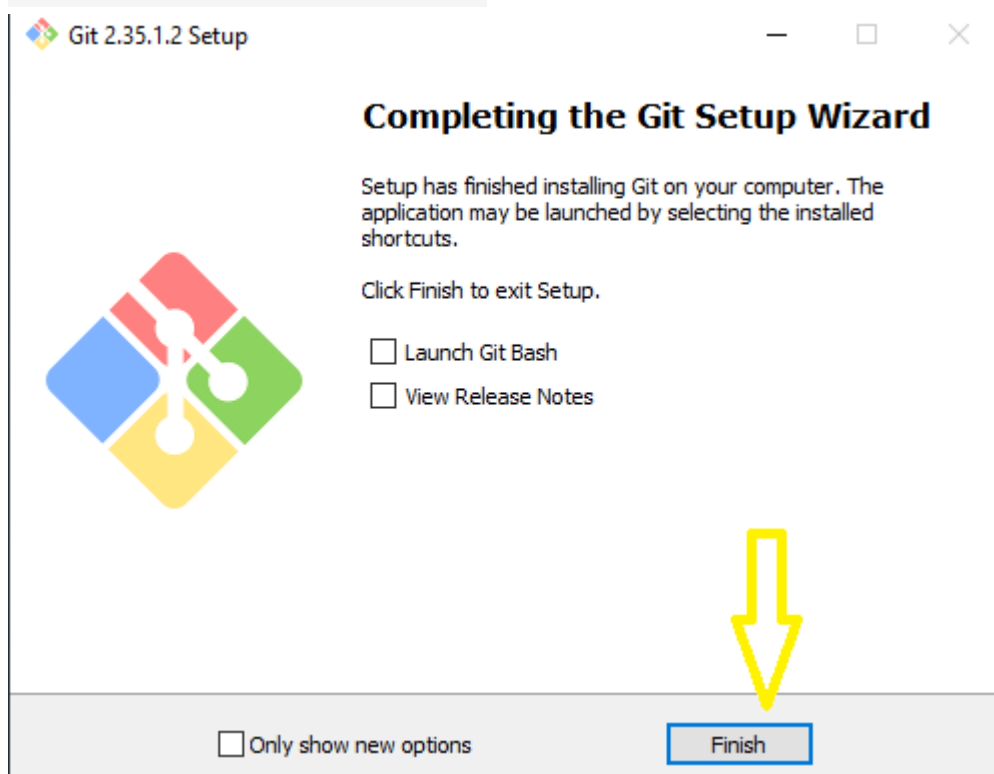
## Step11)



## Step12)



## BOOM HERE WE GO





You can check that Git is installed by simply typing `git --version` in Powershell, git bash.

The page looks like as: -

```
MINGW64:/d/SCM CW
Bat-Computer@DESKTOP-26LA033 MINGW64 /d/SCM CW (master)
$ git --version
git version 2.35.1.windows.2
Bat-Computer@DESKTOP-26LA033 MINGW64 /d/SCM CW (master)
$ |
```

GIT is finally installed on your desktop.

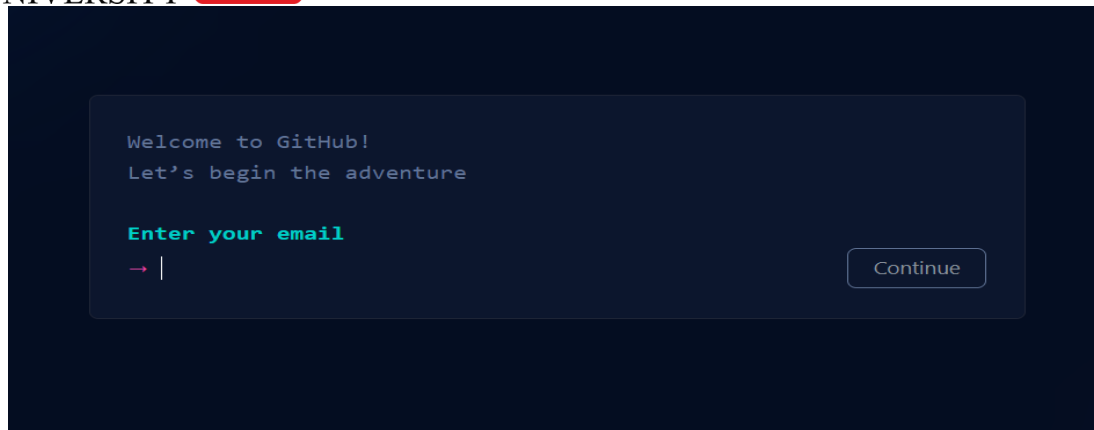
## 1.2 Setting Up GitHub Account

### Step1)

To set up your GitHub account you need to visit <https://github.com/> and click sign-up.

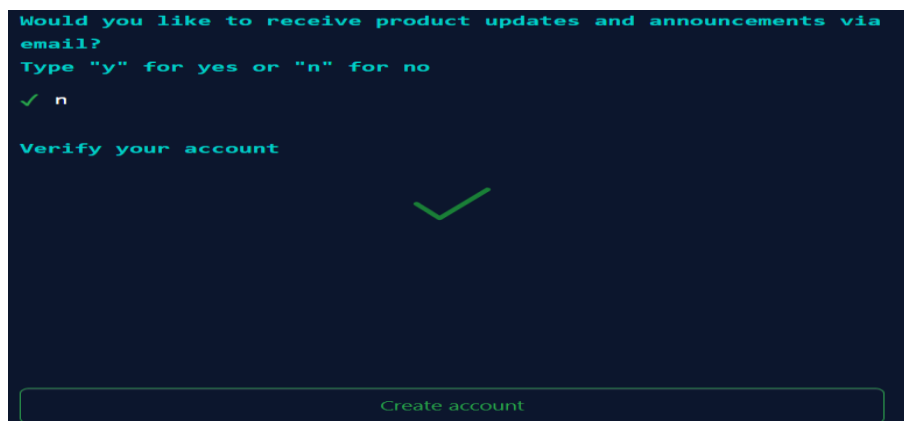
### Step2)

Enter your email, username and desired password.



### Step3)

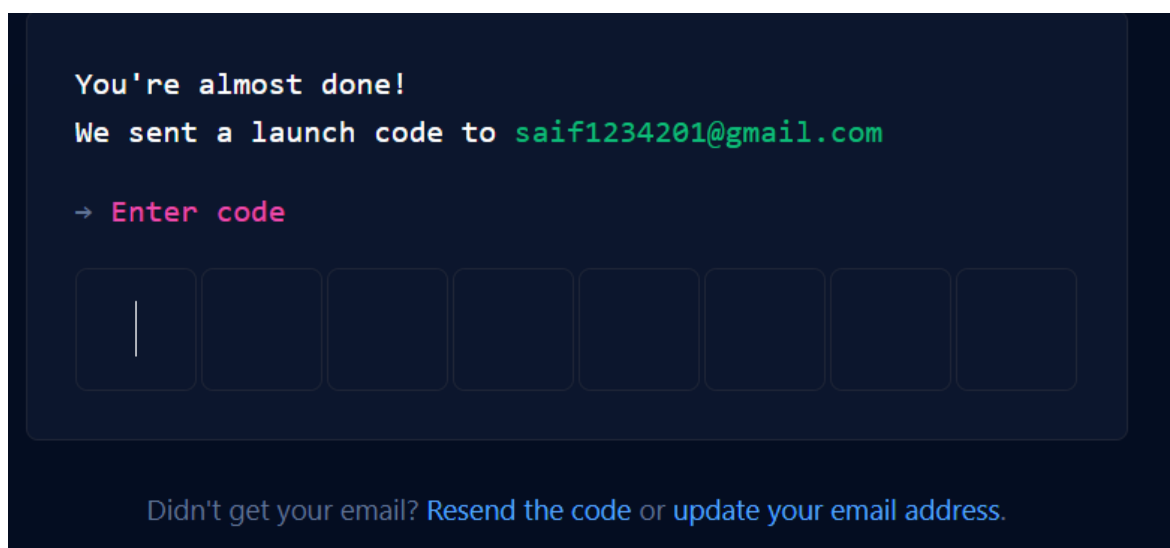
Verify Your Account



### Step4)

Verify Your Email

Enter The Code Received On Your Email



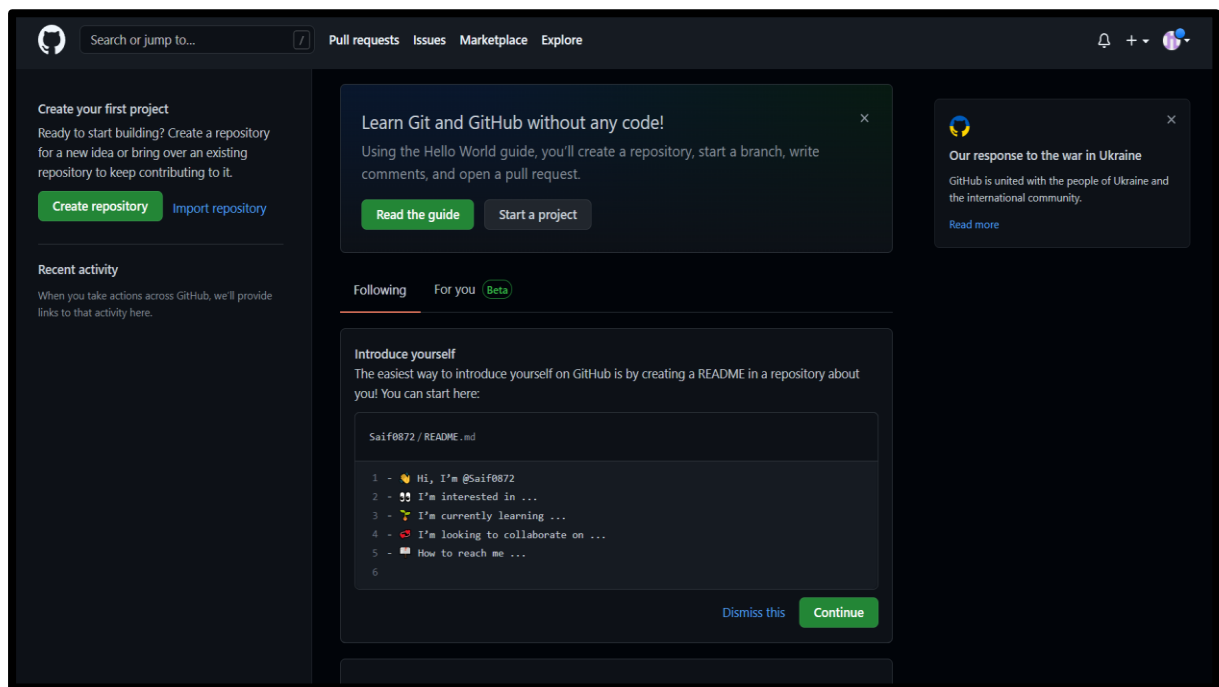


#### Step4)

Select how many memeber with u want to colablrate

And After That

Your account is created



## 1.3 Configuration of Git

### Step1)

You can configure your Git by typing: -

1. Set your username: `git config --global user.name "Your Name"`
2. Set your email address: `git config --global user.email "Your Email"`

The page looks like as: -



```
Bat-Computer@DESKTOP-26LA033 MINGW64 /d/SCM CW (master)
$ git config --global user.name "Saif"

Bat-Computer@DESKTOP-26LA033 MINGW64 /d/SCM CW (master)
$ git config --global user.email "Saif1234201@gmail.com"

Bat-Computer@DESKTOP-26LA033 MINGW64 /d/SCM CW (master)
$ git config --global user.email
Saif1234201@gmail.com

Bat-Computer@DESKTOP-26LA033 MINGW64 /d/SCM CW (master)
$ git config --global user.name
Saif

Bat-Computer@DESKTOP-26LA033 MINGW64 /d/SCM CW (master)
$ |
```

## Step2)

You can check configuration of Git by typing -

1. git config --list

The page looks like as: -

```
$ git config --list
diff.astextplain.textconv=astextplain
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge -- %f
filter.lfs.process=git-lfs filter-process
filter.lfs.required=true
http.sslbackend=openssl
http.sslcainfo=C:/Program Files/Git/mingw64/ssl/certs/ca-bundle.crt
core.autocrlf=true
core.fscache=true
core.symlinks=false
pull.rebase=false
credential.helper=manager-core
credential.https://dev.azure.com.usehttppath=true
init.defaultbranch=master
user.name=Saif
user.email=Saif1234201@gmail.com
core.repositoryformatversion=0
core.filemode=false
core.bare=false
core.logallrefupdates=true
core.symlinks=false
```

2. program to accept N numbers from the user and store them in an array. Then, accept another number from the user and search that using Linear Search.

**program to accept N numbers from the user and store them in an array. Then, accept another number from the user and search that using Linear Search.**

Advantage of version control systems like git is that it can record changes.



'Git log' command is used to display all these changes that were made by the user in the repository. Log is a record of all the previous commits.

To understand Logs we need to get familiar with all the commands that are used in making changes to a repository.

- 1) Repository: A repository is a directory that contains all the project-related data.
- 2) Git init: The git init command is used to create a new blank repository.
- 3) Git status: We can list all the untracked, modified and deleted files using the git status command.
- 4) Git add: Adds all the untracked and modified files to the staging area.
- 5) Git commit: Git commit finalizes the changes in the repository. Every commit is saved in the branch you are working in and can be used to revert back to older versions of the project.

## 2.1 Making GIT Repository

### Step1: GIT INIT

Initializing a new repository, You Can do it by typing: -

1. git init

The page looks like as: -



```
MINGW64:/d/scm project
Bat-Computer@DESKTOP-26LA033 MINGW64 /d/scm project
$ git init
Initialized empty Git repository in D:/scm project/.git/
Bat-Computer@DESKTOP-26LA033 MINGW64 /d/scm project (master)
$ |
```

## Step2:ADDING THE FILES TO THE FOLDER

We can make file using touch command

Just like (This C++ Code)

C++ code is here:

```
//Write a C++ program to search an element in an array using linear search.
//In this C++ program we have to search an element in a given array using linear search algorithm. If given element is present in array then we will print it's index otherwise print a message saying element not found in array.
```

```
//For Example :
//Input Array : [2, 8, 4, 2, 14, 10, 15]
//Element to search : 4
```

```
//Output :
//Element found at index 2
```

```
#include <iostream>
using namespace std;

int main(){
    int input[100], count, i, num;

    cout << "Enter Number of Elements in Array\n";
    cin >> count;

    cout << "Enter " << count << " numbers \n";

    // Read array elements
```



```
for(i = 0; i < count; i++){
    cin >> input[i];
}

cout << "Enter a number to serach in Array\n";
cin >> num;

// search num in inputArray from index 0 to elementCount-1
for(i = 0; i < count; i++){
    if(input[i] == num){
        cout << "Element found at index " << i;
        break;
    }
}

if(i == count){
    cout << "Element Not Present in Input Array\n";
}

return 0;
}
```

The page looks like as: -

The screenshot shows a terminal window titled 'MINGW64: d:/scm project' with the following commands and output:

```
Bat-Computer@DESKTOP-26LA033 MINGW64 /d/scm project (master)
$ git init
Reinitialized existing Git repository in D:/scm project/.git/
Bat-Computer@DESKTOP-26LA033 MINGW64 /d/scm project (master)
$ touch program.cpp
Bat-Computer@DESKTOP-26LA033 MINGW64 /d/scm project (master)
$ |
```

Below the terminal is a file explorer window titled 'Learn (D:) > scm project' showing the following files and folders:

Name	Date modified	Type	Size
.git	13-04-2022 15:50	File folder	
program	13-04-2022 15:50	C++ Source File	0 KB

### Step3:GIT ADD & GIT STATUS

The git add command adds a change in the working directory to the staging area.



You Can do it by typing: -

1. git add
2. git add (current file name)
3. Git status - to see untracked tracked files and other info

The page looks like as: -

```
MINGW64/d/scm project
Bat-Computer@DESKTOP-26LA033 MINGW64 /d/scm project (master)
$ git init
Reinitialized existing Git repository in D:/scm project/.git/
Bat-Computer@DESKTOP-26LA033 MINGW64 /d/scm project (master)
$ touch program.cpp
Bat-Computer@DESKTOP-26LA033 MINGW64 /d/scm project (master)
$ code.
bash: code.: command not found
Bat-Computer@DESKTOP-26LA033 MINGW64 /d/scm project (master)
$ code .
Bat-Computer@DESKTOP-26LA033 MINGW64 /d/scm project (master)
$ git add program.cpp
Bat-Computer@DESKTOP-26LA033 MINGW64 /d/scm project (master)
$ |

Bat-Computer@DESKTOP-26LA033 MINGW64 /d/scm project (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   program.cpp

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

Bat-Computer@DESKTOP-26LA033 MINGW64 /d/scm project (master)
$ |
```

#### Step4:GIT COMMIT



The "commit" command is used to save your changes to the local repository.

You Can do it by typing: -

1. `git commit -m "any text"`

The page looks like as: -

```
Bat-Computer@DESKTOP-26LA033 MINGW64 /d/scm project (master)
$ git commit -m "FirstCommit"
[master (root-commit) dacca8b] FirstCommit
 2 files changed, 33 insertions(+)
 create mode 100644 program.cpp
 create mode 100644 program.exe
Bat-Computer@DESKTOP-26LA033 MINGW64 /d/scm project (master)
$ |
```

### Step5:GIT LOG

Git log will show all the commits made by the author with their time.

After every commit the checksum value (written In yellow color) of the folder changes.

Checksum is used to verify that the data in that file has not been tampered with or manipulated, possibly by a malicious entity.

You Can do it by typing: -

1. `git log`

The page looks like as: -

```
Bat-Computer@DESKTOP-26LA033 MINGW64 /d/scm project (master)
$ git log
commit dacca8b358f4653e0c36489e3238fee6fa1af710 (HEAD -> master)
Author: Saif <Saif1234201@gmail.com>
Date:   Wed Apr 13 21:47:18 2022 +0530

    FirstCommit
Bat-Computer@DESKTOP-26LA033 MINGW64 /d/scm project (master)
$ |
```

### Step6:Create and Visualize Branches

A branch in Git is simply a lightweight movable pointer to one of these commits. The default branch name in Git is master.



## 5. Create and Visualize Branches

### Step1:CHECKING UP THE BRANCHES

You can check which branch you are working in by using the command

1. 'git branch'.

The default branch is always the master branch.

The page looks like as: -

```
HP@Raj MINGW64 ~/OneDrive/Desktop/class matreial/SCM (master)
$ git branch
* master
```

### Step2:CREATING BRANCH AND CHECKING BRANCH

You Can do it by typing: -

1. git branch <BRANCH NAME>

The page looks like as: -

```
Bat-Computer@DESKTOP-26LA033 MINGW64 /d/scm project (master)
$ git branch feature1

Bat-Computer@DESKTOP-26LA033 MINGW64 /d/scm project (mas
ter)
$ GIT BRANCH
fatal: cannot handle BRANCH as a builtin

Bat-Computer@DESKTOP-26LA033 MINGW64 /d/scm project (master)
$ git branch
feature1
* master
snow

Bat-Computer@DESKTOP-26LA033 MINGW64 /d/scm project (master)
$ |
```



### Step3:CHANGING BRANCHES

To switch to the other branch

You Can do it by typing: -

1. git checkout <BRANCH NAME>

The page looks like as: -

```
Bat-Computer@DESKTOP-26LA033 MINGW64 /d/scm project (master)
$ git checkout feature1
Switched to branch 'feature1'

Bat-Computer@DESKTOP-26LA033 MINGW64 /d/scm project (feature1)
$ |
```

### Step4:NOW ADD FEATURE TO THE NEW BRANCH AND COMMIT IT

This looks like as: -

```
Bat-Computer@DESKTOP-26LA033 MINGW64 /d/scm project (feature1)
$ git status
On branch feature1
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:   program.cpp

no changes added to commit (use "git add" and/or "git commit -a")
Bat-Computer@DESKTOP-26LA033 MINGW64 /d/scm project (feature1)
$ git add -A
Bat-Computer@DESKTOP-26LA033 MINGW64 /d/scm project (feature1)
$ git commit -m "Added Question Statement"
[feature1 88559b7] Added Question Statement
```

### Added question statement...

```
//Write a C++ program to search an element in an array using linear search.
//In this C++ program we have to search an element in a given array using linear search algorithm. If given element is
present in array then we will print it's index otherwise print a message saying element not found in array.

//For Example :
//Input Array : [2, 8, 4, 2, 14, 10, 15]
//Element to search : 4

//Output :
//Element found at index 2
```

**Step5:NOW SWITCH BACK TO DEFAULT BRANCH AND CHECK FILE**

This looks like as: -

```
Bat-Computer@DESKTOP-26LA033 MINGW64 /d/scm project (feature1)
$ git checkout master
Switched to branch 'master'

Bat-Computer@DESKTOP-26LA033 MINGW64 /d/scm project (master)
$ |
```

Here when I switch back to master branch question statement gone because I added statement in feature1 one branch and now I am in master branch

```
1 #include <iostream>
2 using namespace std;
3
4 int main(){
5     int input[100], count, i, num;
6 }
```

**Step6:GIT MERGING**

Now you can merge two branches by command.

1. git merge (BRANCH NAME)

If you want to merge a new branch in master branch you need to first checkout into the master branch and then run the command.

The page looks like as: -

```
Bat-Computer@DESKTOP-26LA033 MINGW64 /d/scm project (master)
$ git merge feature1
Updating 7d9026e..88559b7
Fast-forward
 program.cpp | 10 ++++++++
 1 file changed, 10 insertions(+)

Bat-Computer@DESKTOP-26LA033 MINGW64 /d/scm project (master)
$ |
```

Now you can check the files in the master branch.

Both are same now

**Step7:RUNNING GIT LOG**





By running git log command on the master branch you can see all the commits made in master as well as the sample1 branch.

The page looks like as: -

```
bat-Computer@DESKTOP-26LA033 MINGW64 /d/scm project (master)
$ git log
commit 88559b78114e1964f0e624faf7b4576e0317cb93 (HEAD -> master, featur
e1)
Author: Saif <Saif1234201@gmail.com>
Date: Thu Apr 14 09:41:54 2022 +0530

    Added Question Statement

commit 7d9026e429fe09682f54f4db011d0dda871f3ce8
Author: Saif <Saif1234201@gmail.com>
Date: Thu Apr 14 09:23:18 2022 +0530

    second commit

commit dacca8b358f4653e0c36489e3238fee6fa1af710 (snow)
Author: Saif <Saif1234201@gmail.com>
```

## 6. Git Lifecycle Description

There are three stages for git lifecycle:

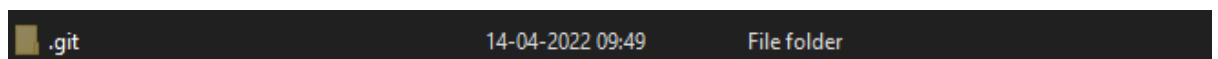
- 1) Working directory
- 2) Staging area
- 3) Git repository

Working Directory:

The working directory is the folder in your local computer where the project files and folders are stored.

The local directory is created by the command 'git init' which creates a '.git' named folder which is used to track the files in the directory.

'git folder' is generally hidden but can be tracked enabling hidden files.



Staging area:



The staging area has those files which are supposed to go to the next commit. Only those files which are needed to go to the next commit stay in the staging area.

You can shift the files to the git repository by using the command

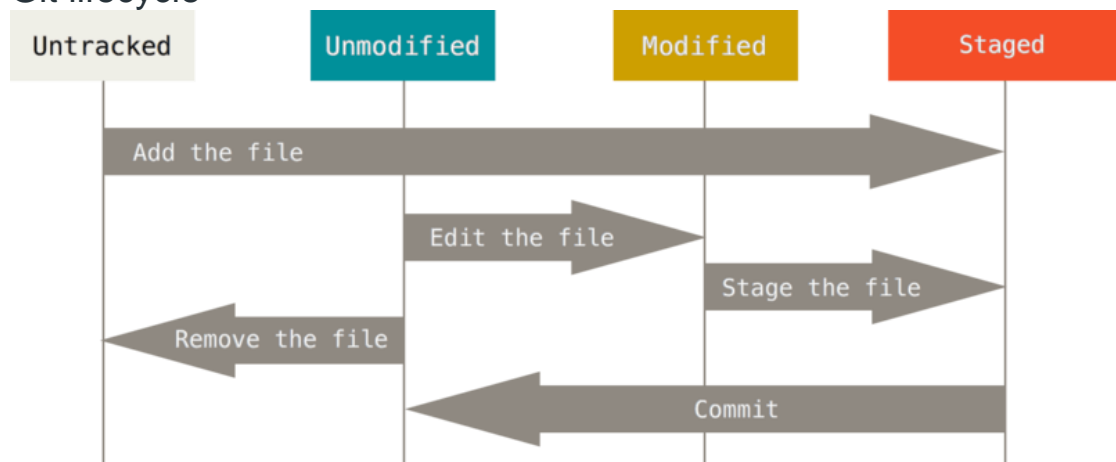
`'git add --a'`.

### Git repository:

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.

You can commit files by using command `'git commit -m`

Git lifecycle -



## 7. UPLOADING DATA ON GITHUB

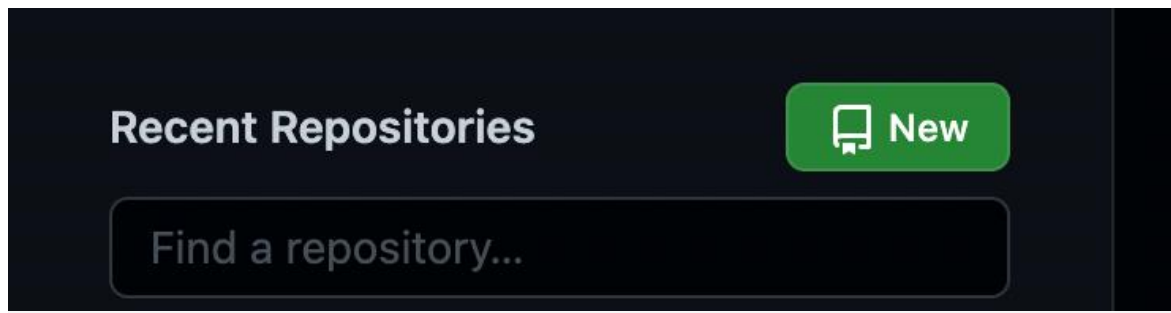
NOTE-



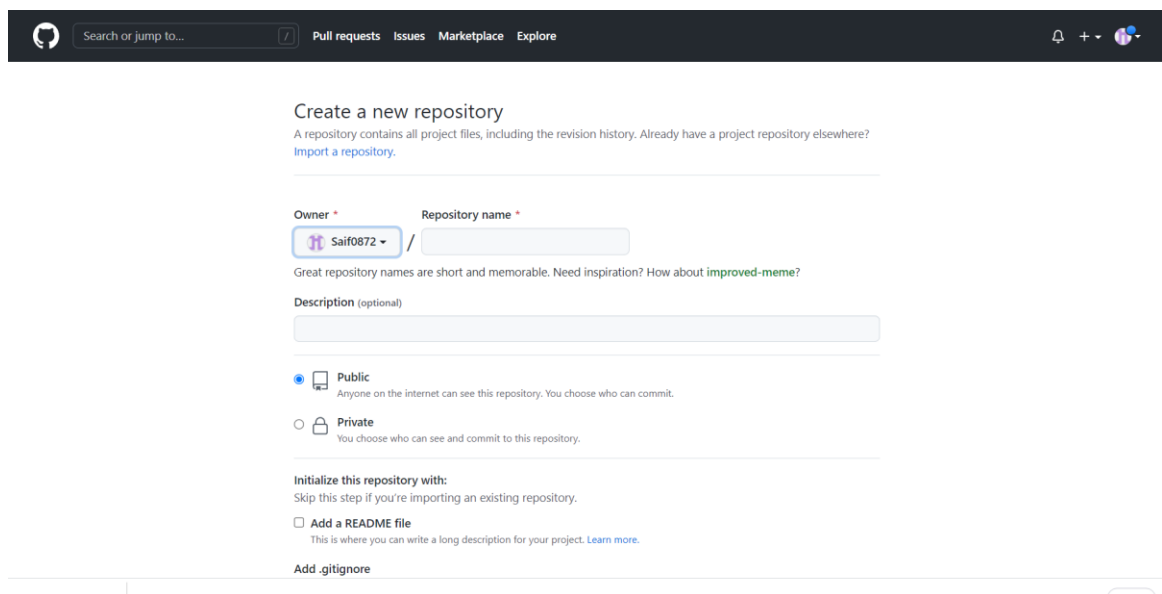
YOU HAVE TO MAKE A REPOSITORY IN GITHUB.

### Step1) CREATING REPOSITORY IN GITHUB

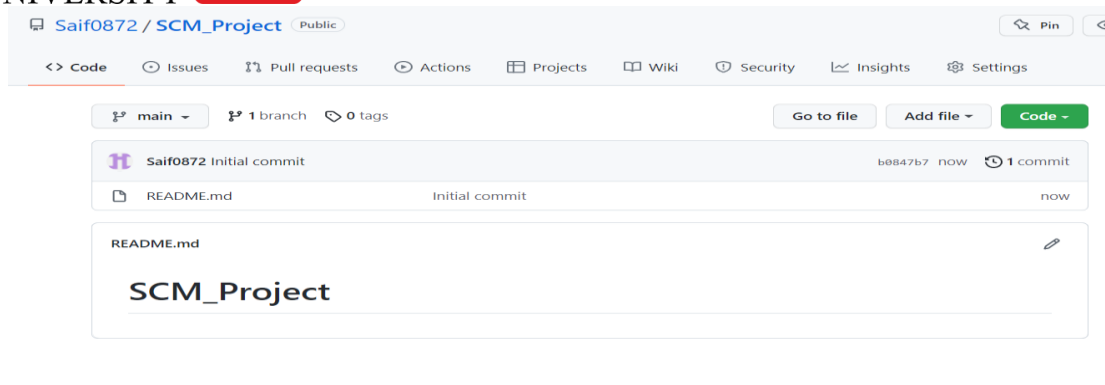
The page looks like as: -



By clicking on new you are able to make a new repository.



Write the repository name and click on next.



Your GITHUB Repository has been created.

## Step2) GIT ADDING REMOTE BRANCH

Git stores a branch as a reference to a commit, and a branch represents the tip of a series of commits.

You Can do it by typing: -

1. `git branch -M main`

The page looks like as: -

```
HP@Raj MINGW64 ~/OneDrive/Desktop/class matreial/SCM (master)
a$ git branch -M main

HP@Raj MINGW64 ~/OneDrive/Desktop/class matreial/SCM (main)
$ |
```

## Step3) GIT ADDING REMOTE ORIGIN

Is a Git repository that's hosted on the Internet

You Can do it by typing: -

1. `git remote add origin (URL)`

The page looks like as: -



```
Bat-Computer@DESKTOP-26LA033 MINGW64 /d/scm project (master)
$ git remote add origin https://github.com/Saif0872/SCM_Project.git

Bat-Computer@DESKTOP-26LA033 MINGW64 /d/scm project (master)
$ git remote
origin

Bat-Computer@DESKTOP-26LA033 MINGW64 /d/scm project (master)
$ |
```

#### Step4) GIT PUSHING

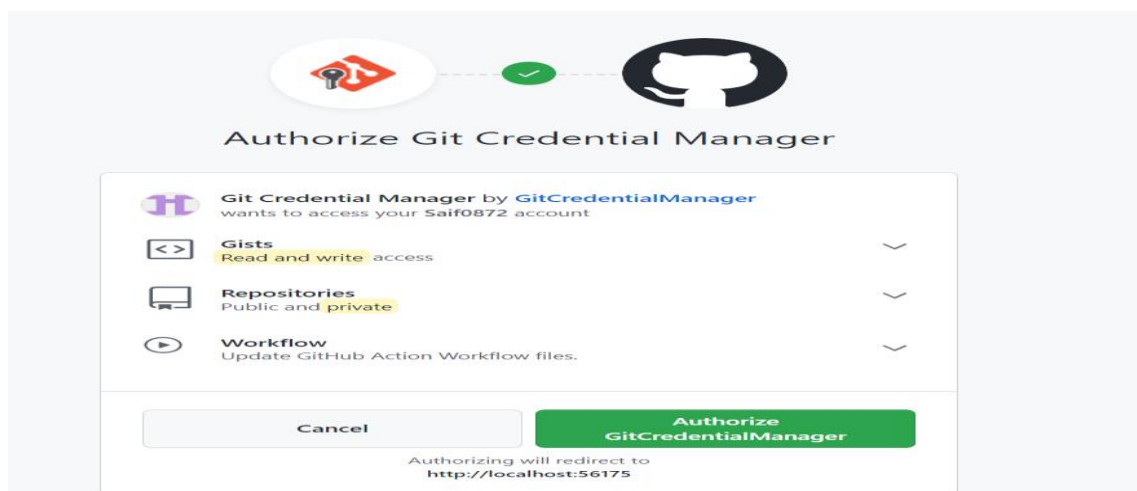
The git push command is used to upload local repository content to a remote repository.

You Can do it by typing: -

1. git push -u origin master

After that a pop up window open for giving access to ur local client to github

The page looks like as: -





## Authentication Succeeded

You may now close this tab and return to the application.

After Authentication Succeeded u will see ur repository pushed

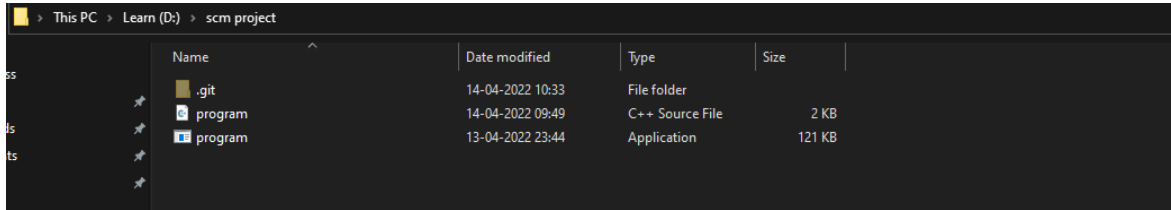
And Your Repository live on github

```
Bat-Computer@DESKTOP-26LA033 MINGW64 /d/NEW SCM PROJECT
(master)
$ git push origin master
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% (3/3), 529 bytes | 264.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'master' on GitHub by
visiting:
remote:   https://github.com/Saif0872/SCM_PROJECT/pull/new/master
remote:
```

## Final Result

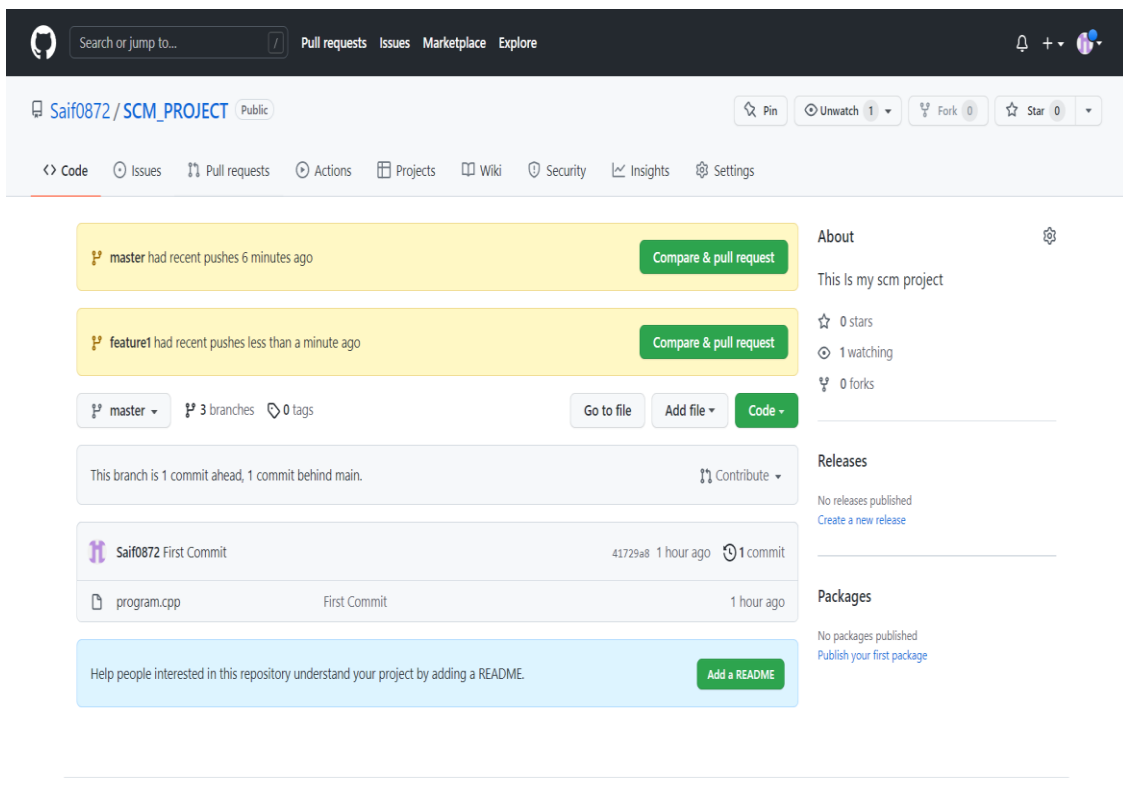
1. Document in your system

The page looks like as: -



## 2. Document in your GITHUB Repository

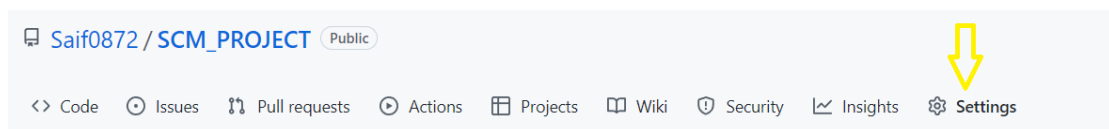
The page looks like as: -



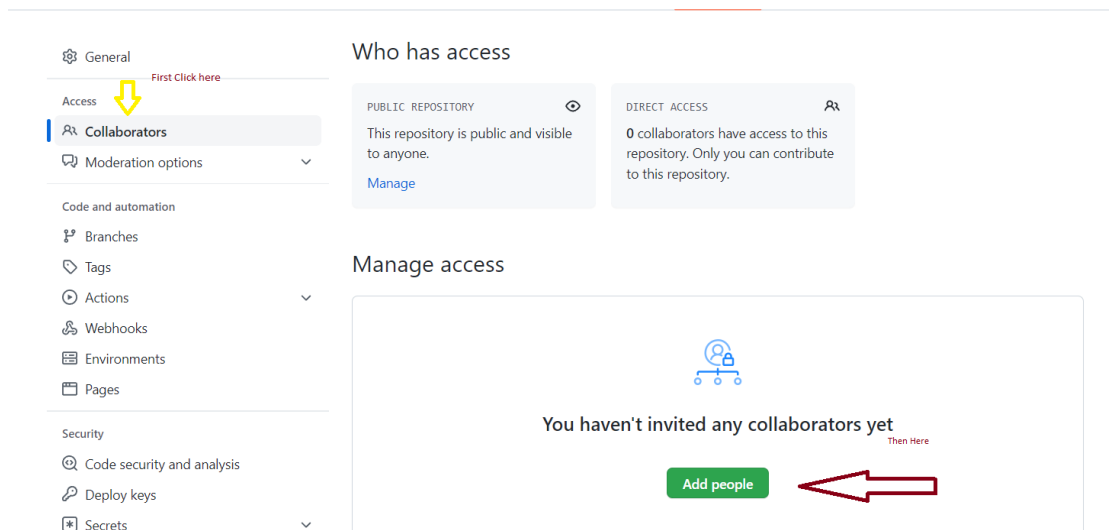


## 2.5 Add collaborators on GitHub Repository:-

1. Ask for the username of the person you're inviting as a collaborator. If they don't have a username yet, they can sign up for GitHub For more information
2. On GitHub.com, navigate to the main page of the repository
3. Under your repository name, click **Settings**.

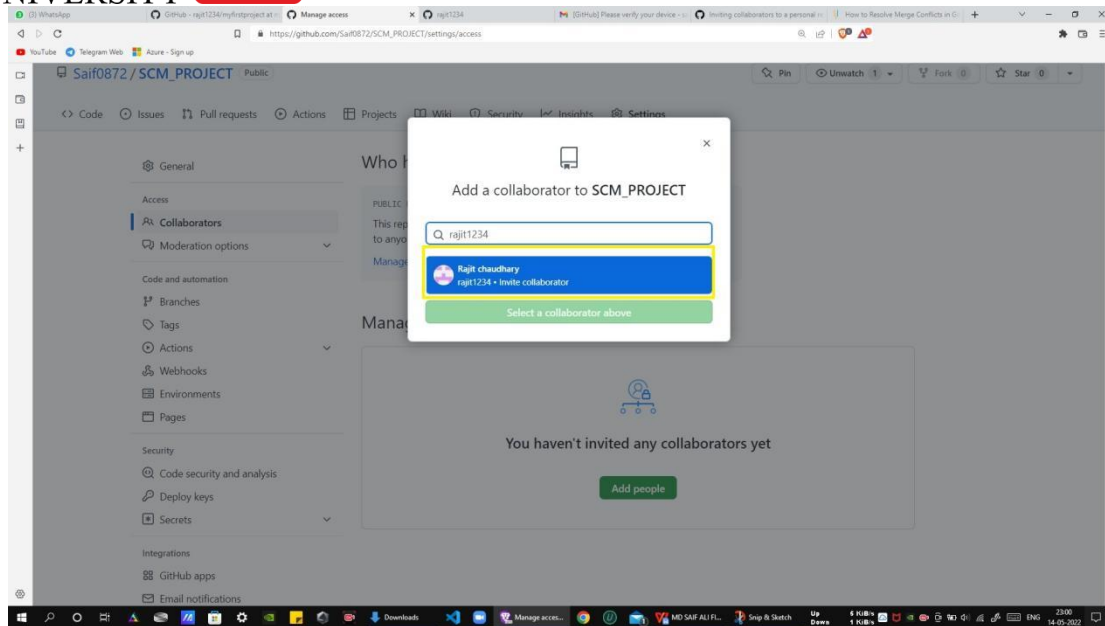


4. In the "Access" section of the sidebar, click **Collaborators & teams**.
5. Click **Invite a collaborator..**

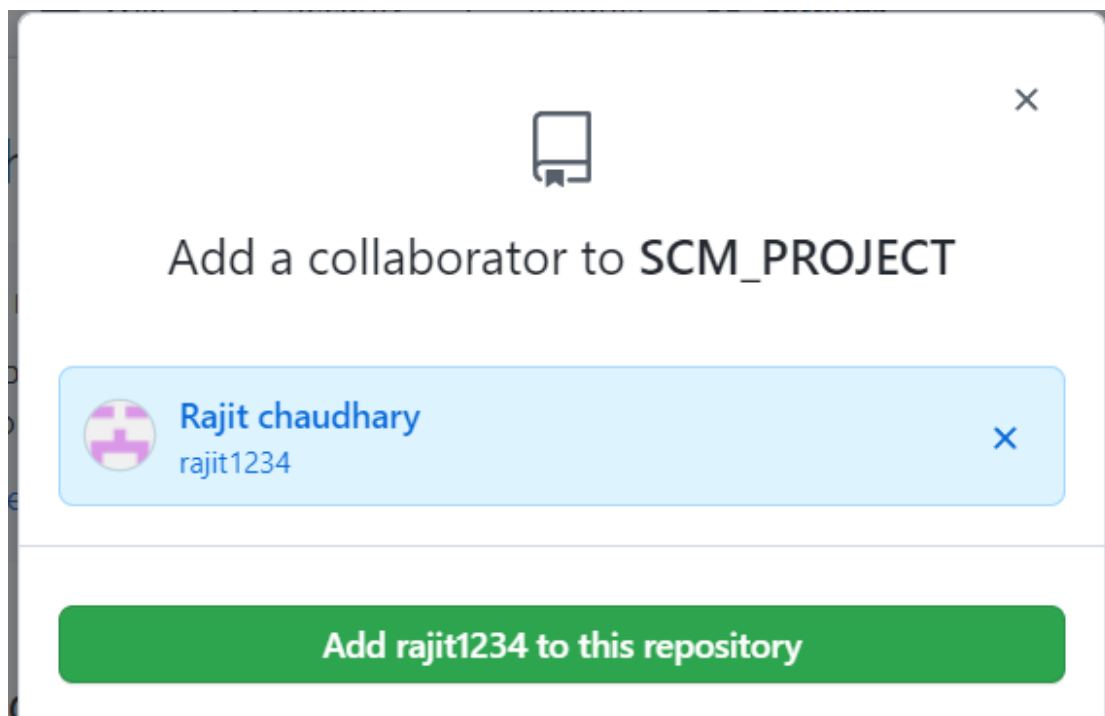


6. In the search field, start typing the name of person you want to invite, then click a name in the list of matches.





7. Click **Add NAME to REPOSITORY**.



8. The user will receive an email inviting them to the repository. Once they accept your invitation, they will have collaborator access to your repository.



## GitHub



@rajit1234 has invited you to collaborate on the  
**rajit1234/myfirstproject** repository

You can [accept](#) or [decline](#) this invitation. You can also head over to <https://github.com/rajit1234/myfirstproject> to check out the repository or visit [@rajit1234](#) to learn a bit more about them.

This invitation will expire in 7 days.

[View invitation](#)

**Note:** This invitation was intended for [saif1234201@gmail.com](mailto:saif1234201@gmail.com). If you were not expecting this invitation, you can ignore this email. If [@rajit1234](#) is sending you too many emails, you can [block them](#) or [report abuse](#).

Getting a 404 error? Make sure you're signed in as [Saif0872](#).

Button not working? Copy and paste this link into your browser:  
<https://github.com/rajit1234/myfirstproject/invitations>

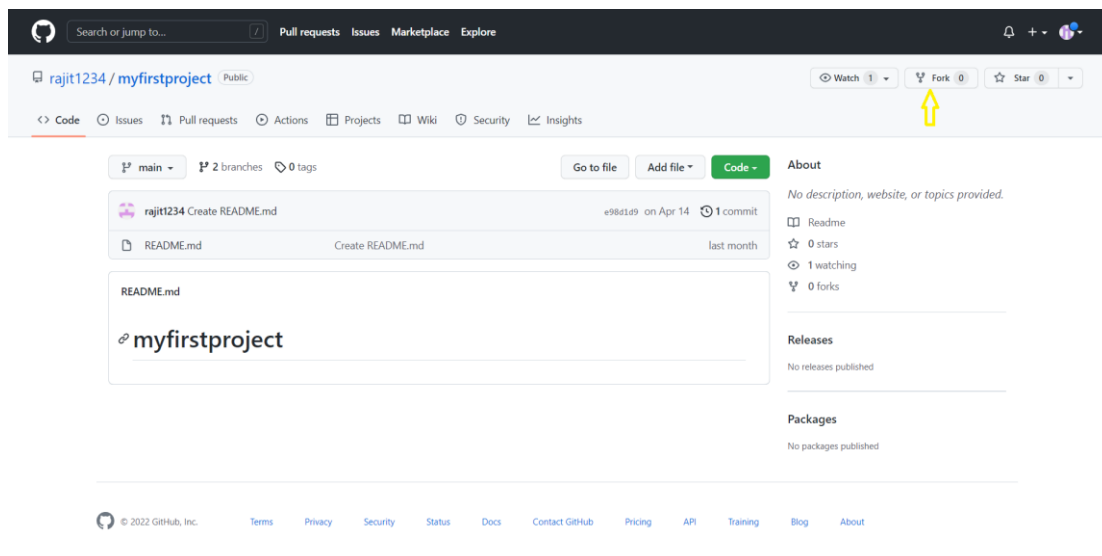
## 2.6 Fork and Commit

The first step is to fork the GitHub repository with which you'd like to work. For example, if you were interested in helping contribute content to the [Open vSwitch](#) web site, which is itself hosted [as a GitHub repository](#), you would first fork it. Forking it is basically making a copy of the repository, but with a link back to the original.



Forking a repository is really straightforward:

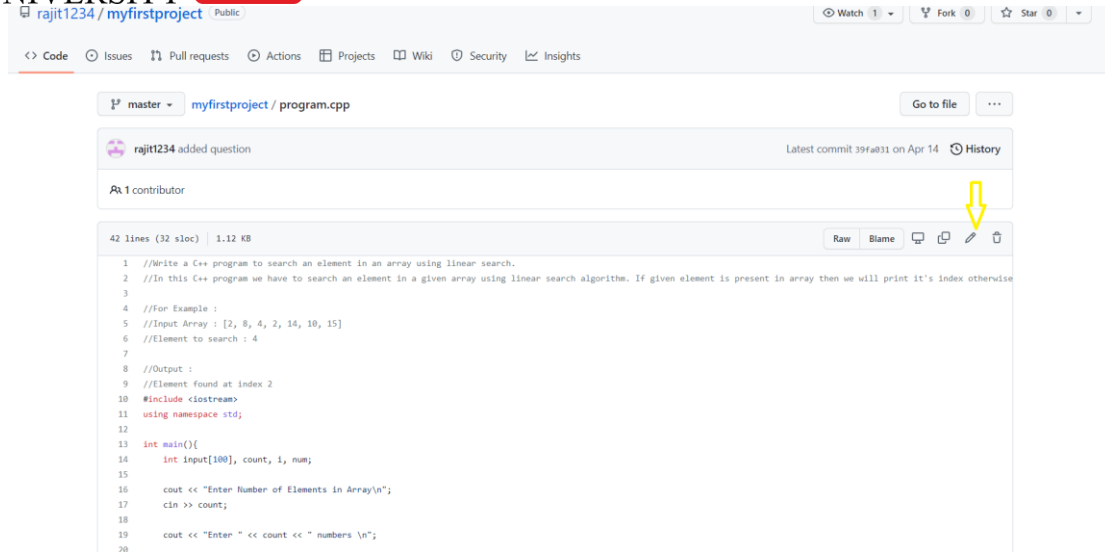
1. Make sure you're logged into GitHub with your account.
2. Find the GitHub repository with which you'd like to work.
3. Click the Fork button on the upper right-hand side of the repository's page.



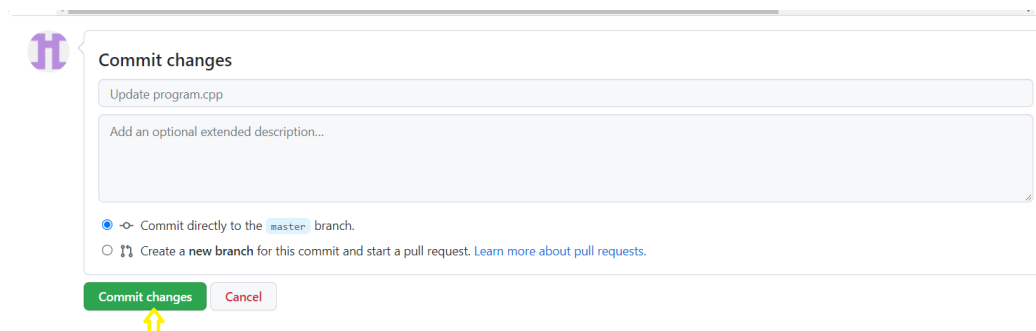
After forking forked repository copied in your account now u can commit changes .... here How...

- 1 . Choose Branch
- 2 . Open file which u want to commit changes
- 3 . Click on pencil icon to edit file

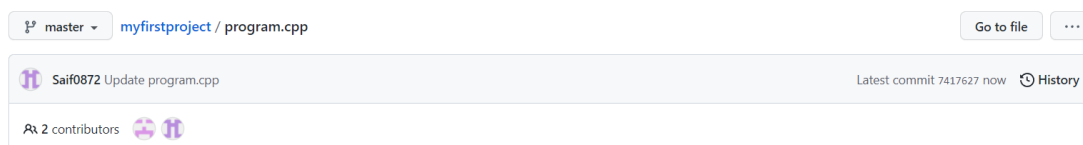
(u need repository access to commit changes otherwise u have to do pull request)



4 .Do Modification and click on commit changes u can also commit in new branch by selecting create a new branch



5 . Then, click on commit change now, Fork commit done !





## 2.7 Merge and Resolve conflicts created due to own activity and collaborators

### Activity:-

#### A - Due to own activity-

To resolve merge conflict first we have to generate conflict for that

Make branch and commit first time

```
Bat-Computer@DESKTOP-26LA033 MINGW64 /d/own-conflict (master)
$ git add .

Bat-Computer@DESKTOP-26LA033 MINGW64 /d/own-conflict (master)
$ git commit -m "commit1"
[master (root-commit) 29f9bf3] commit1
1 file changed, 1 insertion(+)
create mode 100644 new.txt

Bat-Computer@DESKTOP-26LA033 MINGW64 /d/own-conflict (master)
$ |
```

Here I make a txt file and I committed

Now make another branch and change in very same line in that txt file and commit it

I did see below

```
Bat-Computer@DESKTOP-26LA033 MINGW64 /d/own-conflict (master)
$ git branch activity1

Bat-Computer@DESKTOP-26LA033 MINGW64 /d/own-conflict (master)
$ git checkout activity1
Switched to branch 'activity1'

Bat-Computer@DESKTOP-26LA033 MINGW64 /d/own-conflict (activity1)
$ notepad new.txt

Bat-Computer@DESKTOP-26LA033 MINGW64 /d/own-conflict (activity1)
$ git add .

Bat-Computer@DESKTOP-26LA033 MINGW64 /d/own-conflict (activity1)
$ git commit -m "commit2"
[activity1 a6daac2] commit2
1 file changed, 1 insertion(+), 1 deletion(-)

Bat-Computer@DESKTOP-26LA033 MINGW64 /d/own-conflict (activity1)
$ |
```

now Switch back to master and do changes in that line and add commit it

```
Bat-Computer@DESKTOP-26LA033 MINGW64 /d/own-conflict (activity1)
$ git checkout master
Switched to branch 'master'

Bat-Computer@DESKTOP-26LA033 MINGW64 /d/own-conflict (master)
$ notepad new.txt

Bat-Computer@DESKTOP-26LA033 MINGW64 /d/own-conflict (master)
$ git add .

Bat-Computer@DESKTOP-26LA033 MINGW64 /d/own-conflict (master)
$ git commit -m"commit3"
[master 5c3ff6a] commit3
1 file changed, 1 insertion(+), 1 deletion(-)

Bat-Computer@DESKTOP-26LA033 MINGW64 /d/own-conflict (master)
$ |
```

And if we try to merge master and feature 1 conflict will genrate as you can see below

```
MINGW64:/d/own-conflict

Bat-Computer@DESKTOP-26LA033 MINGW64 /d/own-conflict (master)
$ git merge activity1
Auto-merging new.txt
CONFLICT (content): Merge conflict in new.txt
Automatic merge failed; fix conflicts and then commit the result.

Bat-Computer@DESKTOP-26LA033 MINGW64 /d/own-conflict (master|MERGING)
$ .....
```

To solve u can see two method using editor like vs code or we can use mergetool

Here we are going with merge tool

Type git mergetool

Mergetool will configured in your system and open

```
Bat-Computer@DESKTOP-26LA033 MINGW64 /d/own-conflict (master|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:
new.txt

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



```

MINGW64:/d/own-conflict
hello from again master | hello from master branch | hello from activity1 bra
=====
<(09:12 20/05/2022)1,1 All | <(09:12 20/05/2022)1,1 All | <(09:12 20/05/2022)1,1 All
<<<<<<< HEAD
hello from again master branch
=====
hello from activity1 branch
>>>>>>> activity1
new.txt [dos] (09:12 20/05/2022) 1,1 All
"new.txt" [dos] 5L, 104B

```

Then manually edit file file and remove unwanted characters and save it

And finally add commit and now u can merge

```

Bat-Computer@DESKTOP-26LA033 MINGW64 /d/own-conflict (master|MERGING)
$ git add .

Bat-Computer@DESKTOP-26LA033 MINGW64 /d/own-conflict (master|MERGING)
$ git commit
[master 78b2287] Merge branch 'activity1'

Bat-Computer@DESKTOP-26LA033 MINGW64 /d/own-conflict (master)
$ git merge activity1
Already up to date.

Bat-Computer@DESKTOP-26LA033 MINGW64 /d/own-conflict (master)
$ |

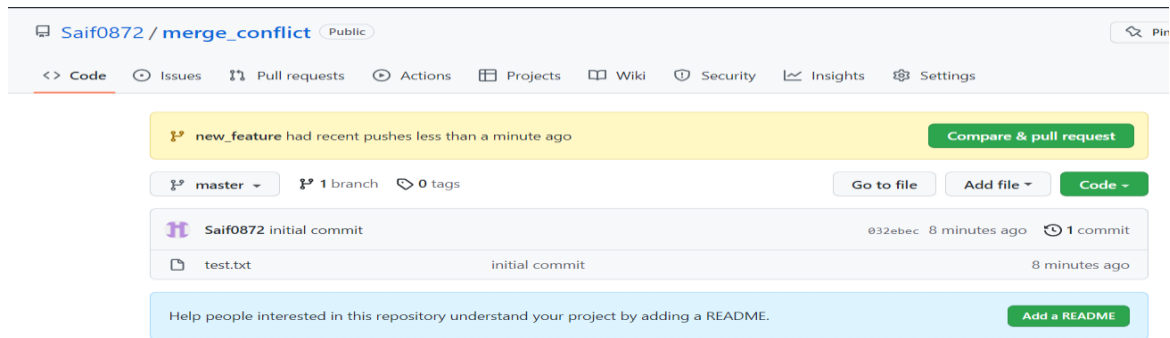
```

## B - Due to Colaborator Activity-



To resolve merge conflict first we have to generate conflict for that

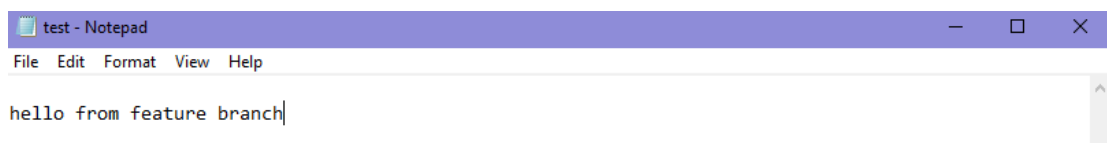
Create a repository and make a file in specific branch and write something in it and do add commit and push it.



Here I pushed a repository called “merge-conflict”

Then we will make a new branch and so modification in one very same line

```
Bat-Computer@DESKTOP-26LA033 MINGW64 /d/conflict (master)
$ git checkout -b new_feature
Switched to a new branch 'new_feature'
```



Initially it was only hello world now in new\_feature branch I modified the same line

And then again do add commit and push

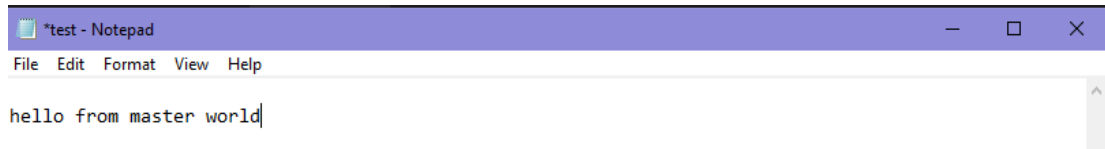
```
Bat-Computer@DESKTOP-26LA033 MINGW64 /d/conflict (new_feature)
$ git push origin new_feature
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Writing objects: 100% (3/3), 269 bytes | 269.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'new_feature' on GitHub by visiting:
remote:   https://github.com/Saif0872/merge_conflict/pull/new/new_feature
remote:
To https://github.com/Saif0872/merge_conflict.git
 * [new branch]      new_feature -> new_feature
```

Now go back to master branch



And again do change in same line

Here I do modification as u can see



And then add commit and push in master branch

Switch to new\_feature

And do pull request using

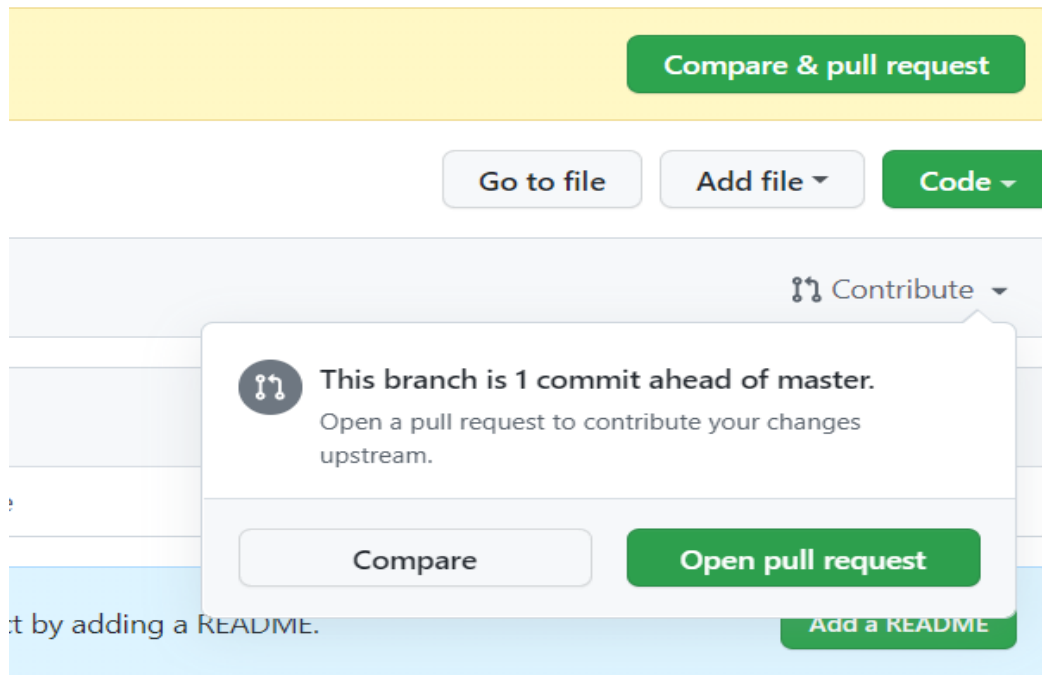
```
git pull origin master
```

And then u see conflict error

```
Bat-Computer@DESKTOP-26LA033 MINGW64 /d/conflict (new_feature)
$ git pull origin master
From https://github.com/Saif0872/merge_conflict
* branch          master      -> FETCH_HEAD
Auto-merging test.txt
CONFLICT (content): Merge conflict in test.txt
Automatic merge failed; fix conflicts and then commit the result.
```

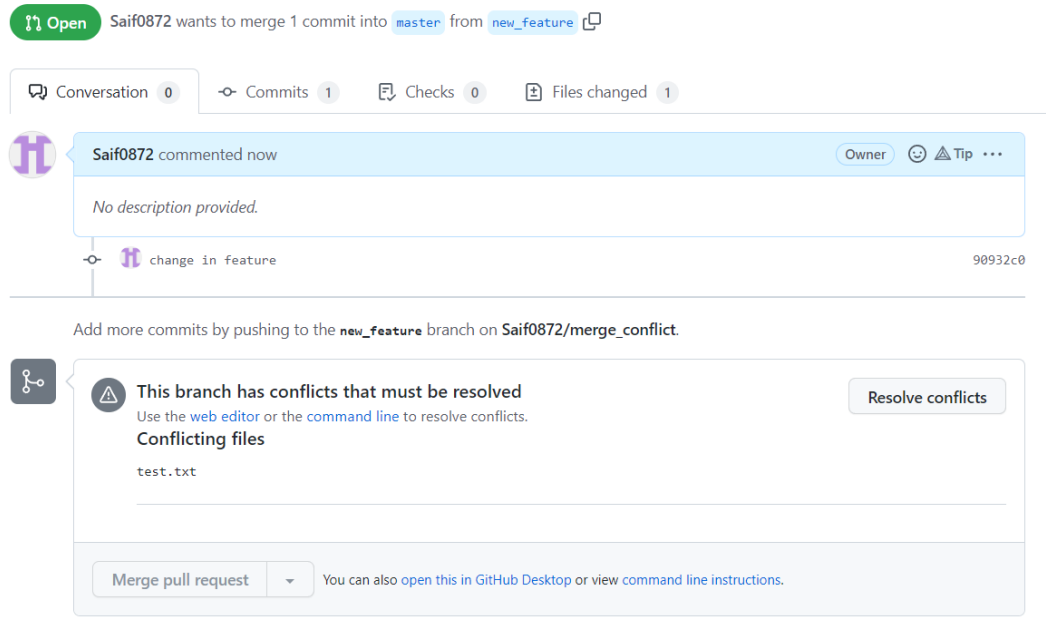
You can also pull request using github mentioned below:

To pull request github repository and raise pull request by clicking on contribute button as u can see below:



Then click on create pull request button and you see conflict generated

## change in feature #1



Now we have to solve this conflict

If u check status u will see unmerged path:



```
Bat-Computer@DESKTOP-26LA033 MINGW64 /d/conflict (new_feature|MERGING)
$ git status
On branch new_feature
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:   test.txt

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

Switch back to new\_feature branch because conflict is in new\_feature branch

Using git checkout new\_feature

Now there are three way to solve conflict

1. Using github
2. Using normal editor
3. And u can use mergetool via git

Here u are going to solve using mergetool via git

For that type

mergetool

```
Bat-Computer@DESKTOP-26LA033 MINGW64 /d/conflict (new_feature|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:
test.txt

Normal merge conflict for 'test.txt':
  {local}: modified file
  {remote}: modified file
Hit return to start merge resolution tool (vimdiff):
4 files to edit
  0 [sig] bash 1492! sigpacket::process: Suppressing signal 18 to win32 proc
ess (pid 1992)

Bat-Computer@DESKTOP-26LA033 MINGW64 /d/conflict (new_feature|MERGING)
```

After mergetool successfully configured

A pop open as you can see below



```

MINGW64:/d/conflict
hello from feature branch  hello World  hello from master world
=====
<(08:42 15/05/2022)1,1 A11 <(08:42 15/05/2022)1,1 A11 <(08:42 15/05/2022)1,1 A11
<<<<<<< HEAD
hello from feature branch
=====
hello from master world
>>>>>>> 825531615558888575bd3c416ad086379ac79b2e

test.txt [dos] (08:42 15/05/2022) 1,1 A11
"test.txt" [dos] 5L, 125B

```

Now decide which modification u want to keep

Press i for remove extra unwanted character

And edit and press esc (for exiting insert mode)

Then :wq for exiting mergetool

After exiting

Do add and commit and push

Keep in mind we don't have to write commit message because it automatically add a commit message

```

Bat-Computer@DESKTOP-26LA033 MINGW64 /d/conflict (new_feature|MERGING)
$ git add .

Bat-Computer@DESKTOP-26LA033 MINGW64 /d/conflict (new_feature|MERGING)
$ git commit
[new_feature 6f2c2fa] Merge branch 'master' of https://github.com/Saif0872/merge_conflict into new_feature

Bat-Computer@DESKTOP-26LA033 MINGW64 /d/conflict (new_feature)
$ git push origin new_feature
Enumerating objects: 12, done.
Counting objects: 100% (12/12), done.
Delta compression using up to 8 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (8/8), 932 bytes | 310.00 KiB/s, done.
Total 8 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), done.
To https://github.com/Saif0872/merge_conflict.git
 90932c0..6f2c2fa new_feature -> new_feature

Bat-Computer@DESKTOP-26LA033 MINGW64 /d/conflict (new_feature)
$

```

git log and here you can see commit message automatically added

```
Bat-Computer@DESKTOP-26LA033 MINGW64 /d/conflict (new_feature)
$ git log
commit 6f2c2fac2fb5e77482aeb1a7a66de43d63dd539b (HEAD -> new_feature, origin/new_feature)
Merge: 90932c0 8255316
Author: Saif0872 <saif1234201@gmail.com>
Date: Sun May 15 08:45:56 2022 +0530

    Merge branch 'master' of https://github.com/Saif0872/merge_conflict into new_feature


commit 82553161555888575bd3c416ad086379ac79b2e (origin/master, master)
Author: Saif0872 <saif1234201@gmail.com>
Date: Sun May 15 08:35:23 2022 +0530


    change in master
```

Now our conflict Resolved u can also check on github


This branch has no conflict with the base branch

change in feature #1





 Open

 Saif0872 wants to merge 2 commits into `master` from `new_feature` 


Conversation 0 Commits 2 Checks 0 Files changed 1


 Saif0872 commented 10 minutes ago Owner Tip ...


No description provided.

-   change in feature 90932c0
-   Merge branch 'master' of https://github.com/Saif0872/merge\_conflict i... 6f2c2fa

Add more commits by pushing to the `new_feature` branch on Saif0872/merge\_conflict.

 Continuous integration has not been set up  
GitHub Actions and several other apps can be used to automatically catch bugs and enforce style.

 This branch has no conflicts with the base branch  
Merging can be performed automatically.

Merge pull request  You can also open this in GitHub Desktop or view command line instructions.



## 2.8 Reset:-

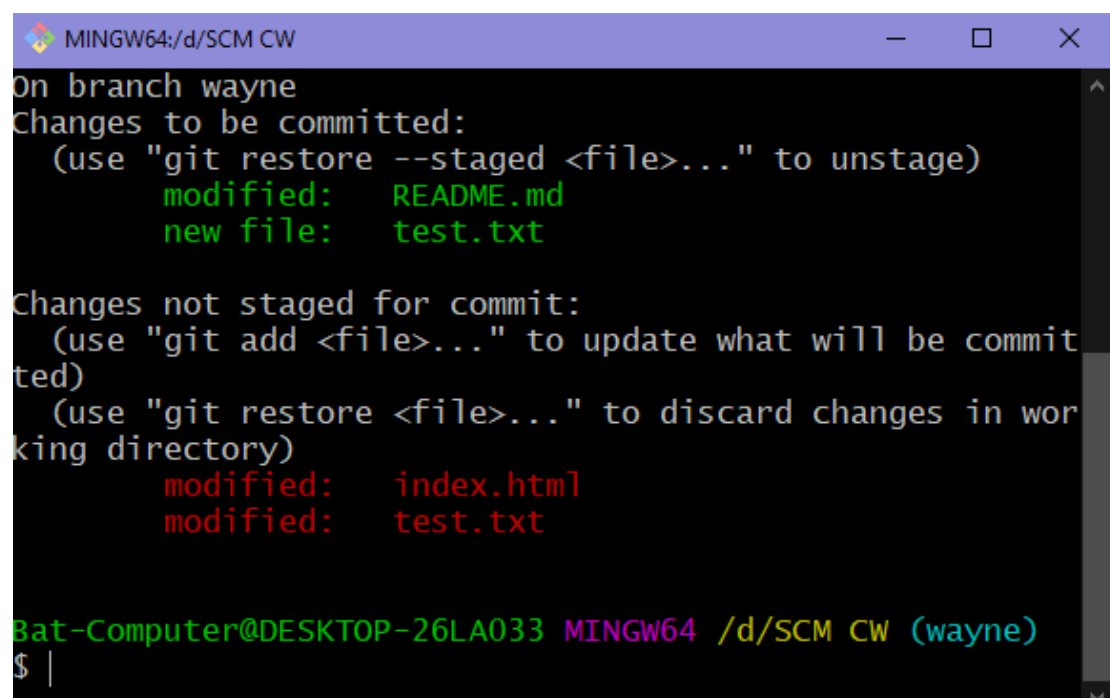
When working on a team project, it is quite common for developers to create branches, add files and stage them for commits when they are ready.

However, in some cases, you might realize that the changes that you made were not that good.

You modified some files, added and deleted a lot of lines from your files, but you want to go back.

In short, you want to revert the changes that you just made and go back to the files that you had.

This technique is called “reset to HEAD” and it is quite a powerful tool for developers to use.



```
MINGW64:/d/SCM CW
On branch wayne
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   README.md
        new file:   test.txt

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:   index.html
        modified:   test.txt

Bat-Computer@DESKTOP-26LA033 MINGW64 /d/SCM CW (wayne)
$ |
```

Here I Modified index.html and made new test.txt file

I Addedd two lines in test.txt as u can see



```
test - Notepad
File Edit Format View Help
hello world
```

Then after add and commit

```
Bat-Computer@DESKTOP-26LA033 MINGW64 /d/SCM CW (wayne)
$ git add .

Bat-Computer@DESKTOP-26LA033 MINGW64 /d/SCM CW (wayne)
$ |
```

```
MINGW64:/d/SCM CW
$ git commit -m "Added Some text "
[wayne f80c529] Added Some text
1 file changed, 2 insertions(+), 1 deletion(-)

Bat-Computer@DESKTOP-26LA033 MINGW64 /d/SCM CW (wayne)
$ git reset --soft HEAD~1

Bat-Computer@DESKTOP-26LA033 MINGW64 /d/SCM CW (wayne)
$ git status
On branch wayne
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   test.txt

Bat-Computer@DESKTOP-26LA033 MINGW64 /d/SCM CW (wayne)
$ |
```

Run this - **git reset --soft HEAD~1**

Now ur file is back to staging area (Modified) without removing changes u made in file (for example I added lines “bye world” in test.txt) is still in file..

```
test - Notepad
File Edit Format View Help
hello World
bye World
```



If we want to delete that line and also remove from staging area (modified)

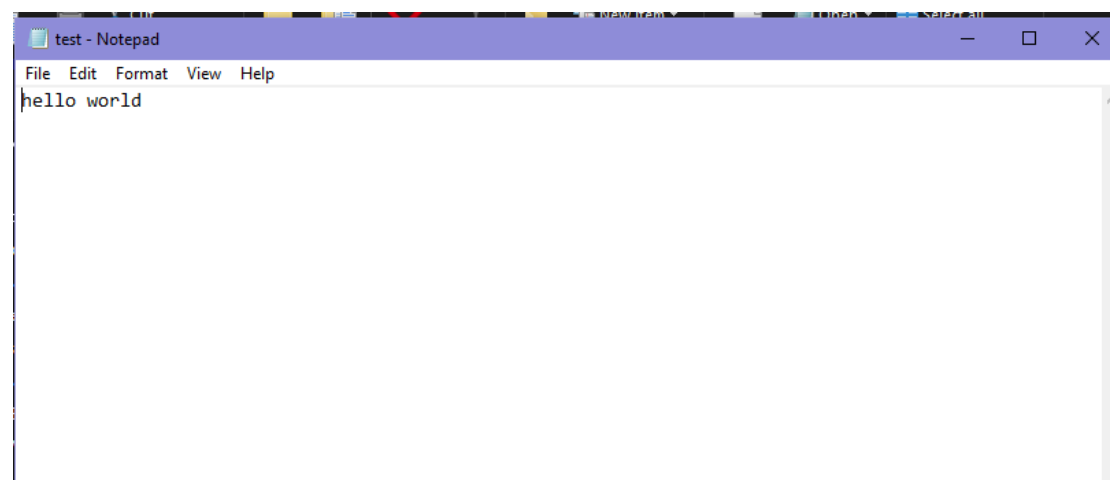
We have to do hard reset

Run this - **git reset --hard HEAD~1**

```
Bat-Computer@DESKTOP-26LA033 MINGW64 /d/SCM CW (wayne)
$ git reset --hard HEAD~1
HEAD is now at 55914cd Added test

Bat-Computer@DESKTOP-26LA033 MINGW64 /d/SCM CW (wayne)
$ |
```

Now added lines “bye World” removed as u can see



## 2.9Git Ignore:-

Often, when working on a project that uses Git, you'll want to exclude specific files or directories from being pushed to the remote repository. This is where `.gitignore` file comes in handy.



The `.gitignore` file specifies what untracked files Git should ignore.

First Create a gitignore file by typing

`touch .gitignore`

```
Bat-Computer@DESKTOP-26LA033 MINGW64 /d/SCM CW (wayne)
$ touch .gitignore

Bat-Computer@DESKTOP-26LA033 MINGW64 /d/SCM CW (wayne)
$ |
```

Open gitignore and add file name which u don't want to track for example log file bin file etc

U can also do `*<extention>` to exclude all file with given extention

For being tracked



Here I added mtlog.log in gitignore to exclude log files..

It will not affect other files

Run git status to check

```
Bat-Computer@DESKTOP-26LA033 MINGW64 /d/SCM (wayne)
$ git status
On branch wayne
nothing to commit, working tree clean

Bat-Computer@DESKTOP-26LA033 MINGW64 /d/SCM (wayne)
$ |
```



## 3.0 Create a distributed Repository and add members in project team

**Aim: Create a distributed Repository and add members in project team**

### **Distributed Repository:**

Now let's discuss how to work with the distributed approach.

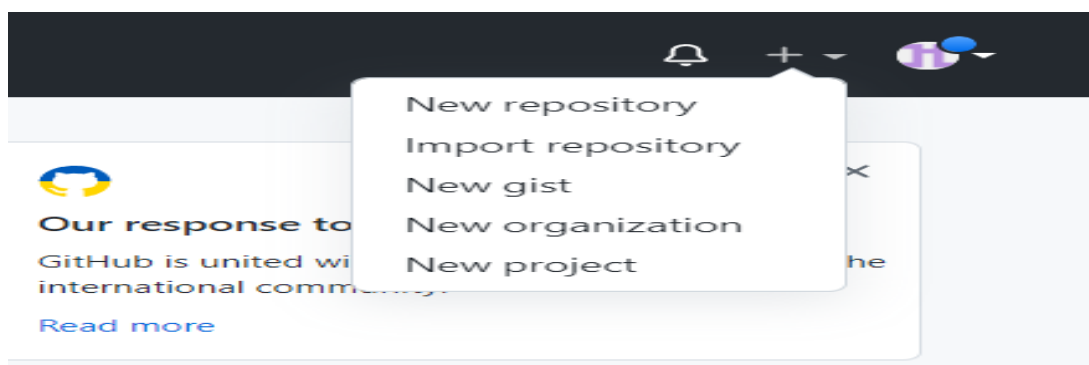
While in centralized approach, everyone on team cloned the single central repository, here in distributed approach everyone clones 'a copy of the source repository' also known as 'forking'.

In this approach, all the collaborators will 'fork' the source repo.

Upon forking every member gets their own copy of the source repository in their GitHub account. Collaborators can then go ahead and clone their 'forked repository' on their local machines.

### Steps of creating Distributed Repository:

we will create a new repository and invite a collaborator to our repository by sending an invitation. A detailed procedure on how to invite a collaborator to your Github repository is mentioned below.





## Specify the Name of the Project

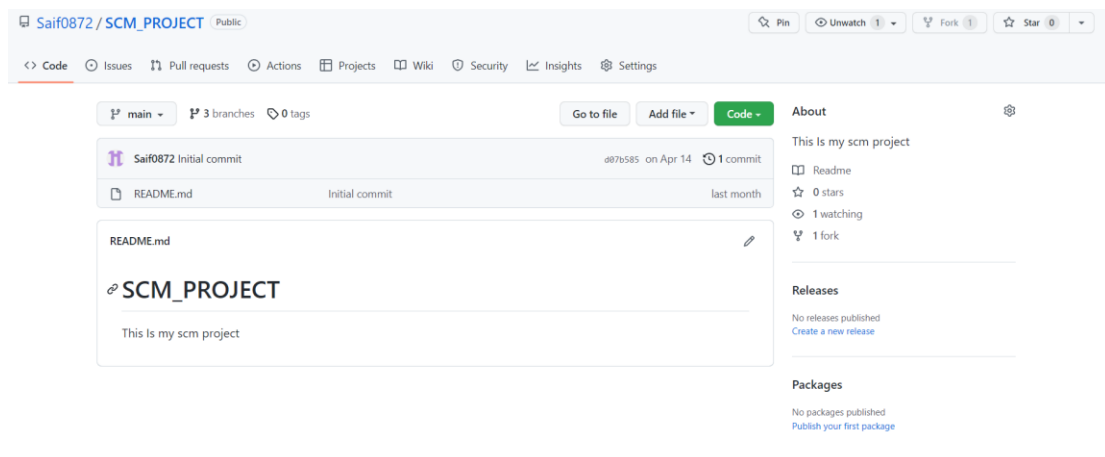
Owner \* Repository name \*

Saif0872 / SCM\_PROJECTT ✓

Great repository names are available. SCM\_PROJECTT is available. Need inspiration? How about [didactic-octo-sniffle?](#)

Description (optional)

Now you can see the file created on that name with a default  
readme file



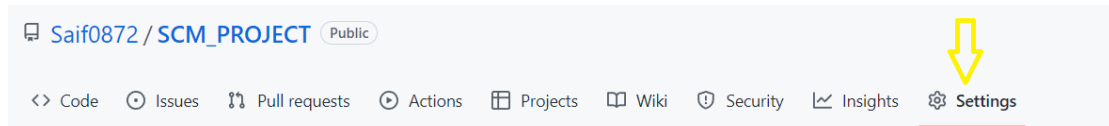
**GitHub collaboration** is a space where you can invite another developer to your repository and work together at an organization level, splitting the task and 2nd person will have the rights to add or merge files to the main repository without further permission. Also, refer to this Github Beginner guide. Let's look into how to add collaborators into Github.

Ask for the username of the person you're inviting as a collaborator. If they don't have a username yet, they can sign up for Github For more information

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

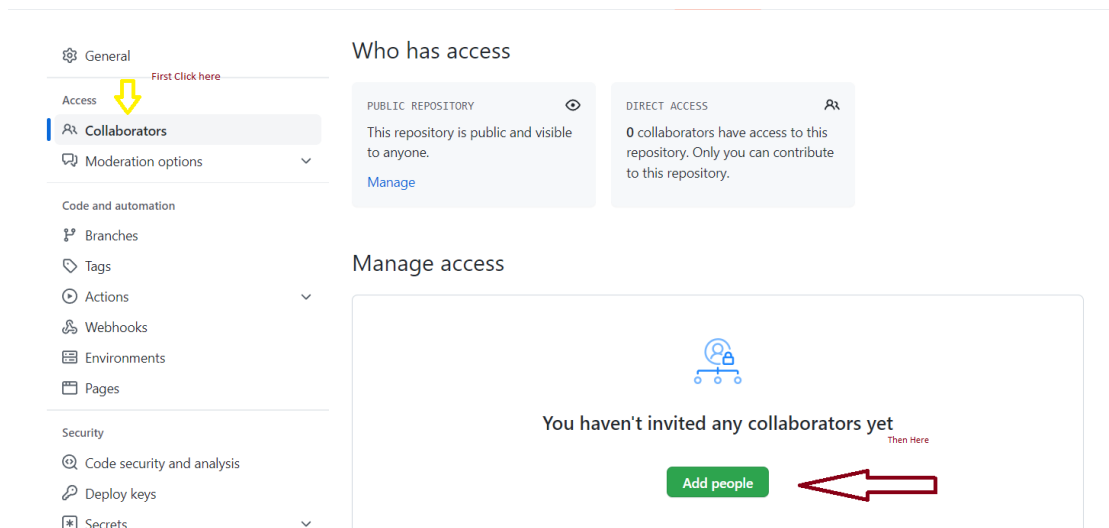


Under your repository name, click **Settings**.

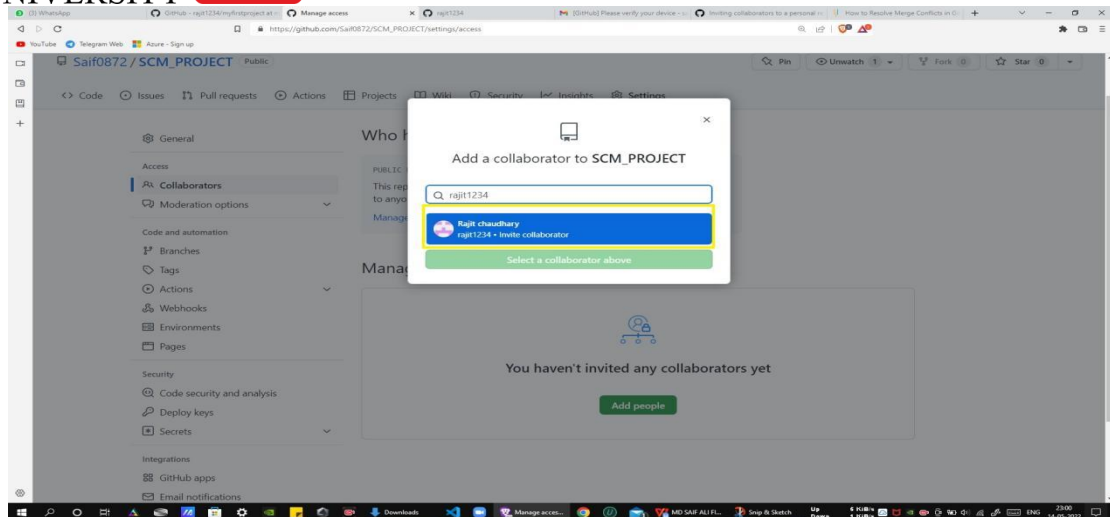


In the "Access" section of the sidebar, click **Collaborators & teams**.

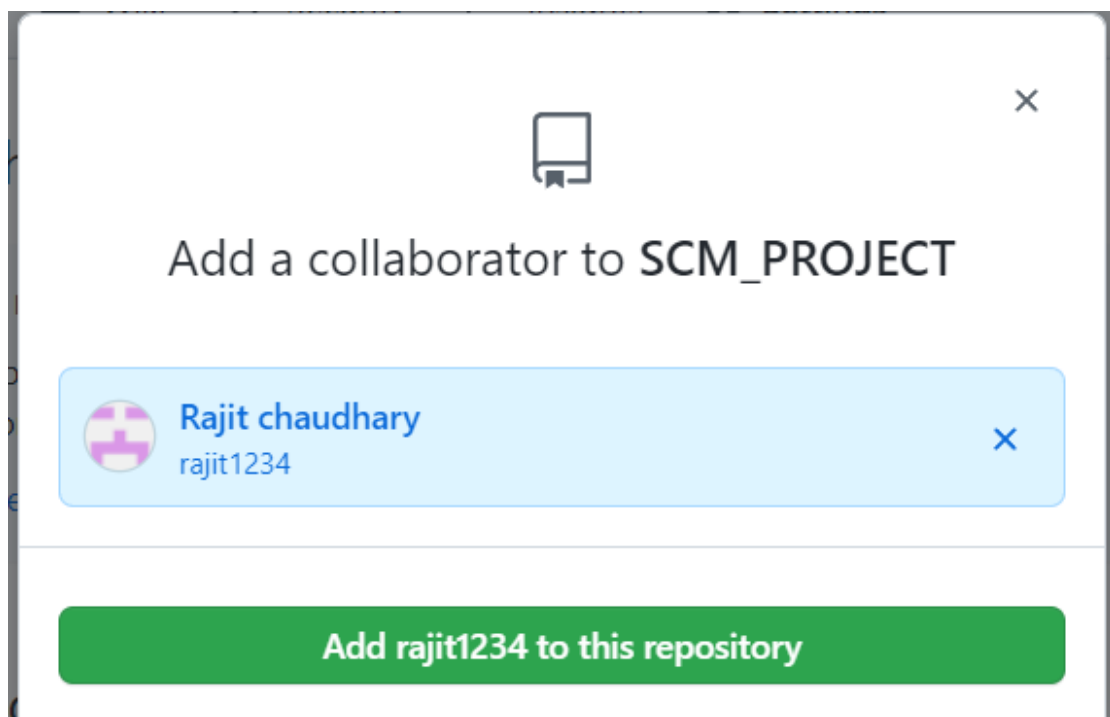
Click **Invite a collaborator**.



In the search field, start typing the name of person you want to invite, then click a name in the list of matches.



Click **Add NAME to REPOSITORY**.



The user will receive an email inviting them to the repository. Once they accept your invitation, they will have collaborator access to your repository.



## GitHub



@rajit1234 has invited you to collaborate on the  
**rajit1234/myfirstproject** repository

You can [accept or decline](#) this invitation. You can also head over to <https://github.com/rajit1234/myfirstproject> to check out the repository or visit [@rajit1234](#) to learn a bit more about them.

This invitation will expire in 7 days.

[View invitation](#)

**Note:** This invitation was intended for [saif1234201@gmail.com](mailto:saif1234201@gmail.com). If you were not expecting this invitation, you can ignore this email. If [@rajit1234](#) is sending you too many emails, you can [block them](#) or [report abuse](#).

Getting a 404 error? Make sure you're signed in as [Saif0872](#).

Button not working? Copy and paste this link into your browser:  
<https://github.com/rajit1234/myfirstproject/invitations>



### 3.1 Open and close a pull request.

click on the Fork button in the top-right corner.

The copy includes all the code, branches, and commits

from the original repo. Next,

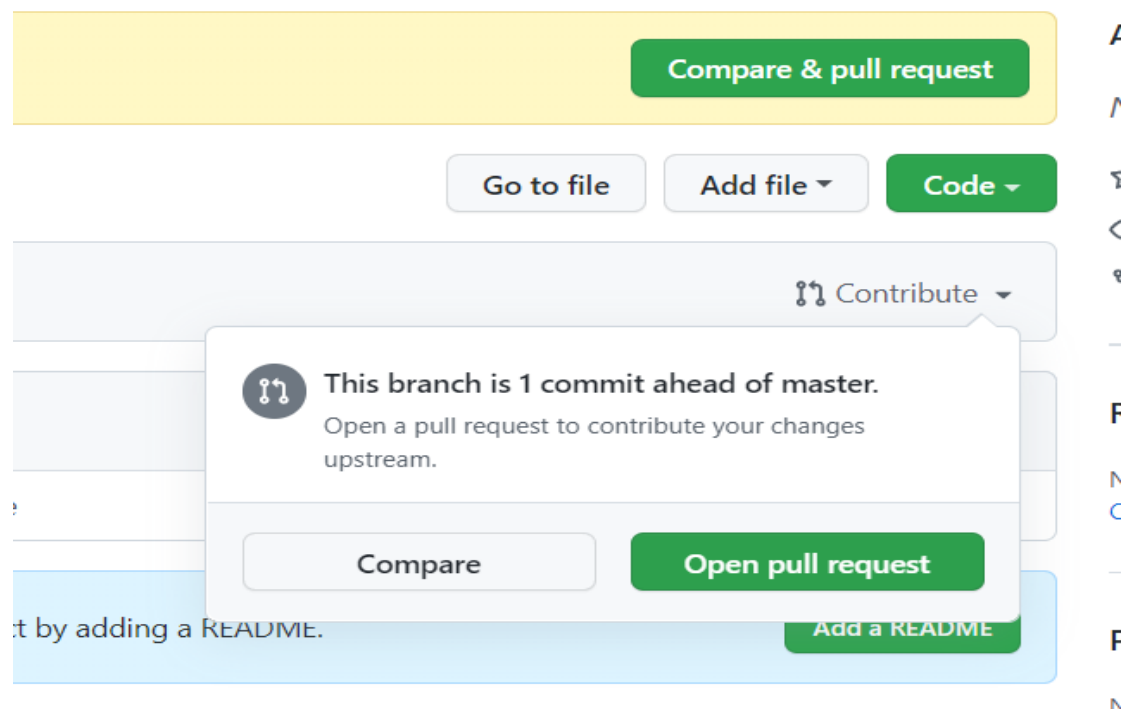
2. clone the repository and Create a new branch .

3. Now you can make changes to the code. The following code creates a new branch, makes an arbitrary change, and pushes it to new\_branch:

4. Once you push the changes to your repo, the Compare & pull request button will appear in GitHub.

You can pull request using github mentioned below:

To pull request github repository and raise pull request by clicking on contribute button as u can see below:



Then click on create pull request button

## change in feature #1

**Open** Saif0872 wants to merge 2 commits into `master` from `new_feature`

Conversation 0 Commits 2 Checks 0 Files changed 1

**Saif0872** commented 10 minutes ago Owner Tip ...

No description provided.

- change in feature 90932c0
- Merge branch 'master' of https://github.com/Saif0872/merge\_conflict i... 6f2c2fa

Add more commits by pushing to the `new_feature` branch on Saif0872/merge\_conflict.

**Continuous integration has not been set up**  
GitHub Actions and several other apps can be used to automatically catch bugs and enforce style.

**✓ This branch has no conflicts with the base branch**  
Merging can be performed automatically.

**Merge pull request** You can also [open this in GitHub Desktop](#) or view [command line instructions](#).

Now u can alos Merge pull request

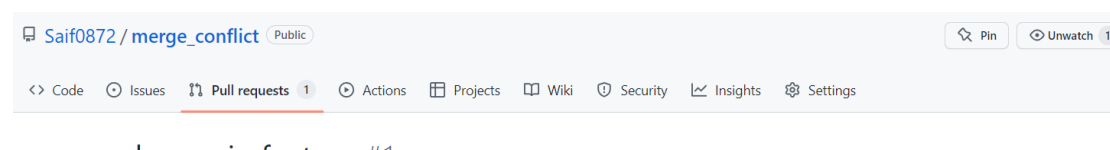
## close a pull request

You may choose to close a pull request without merging it into

the upstream branch. This can be handy if the changes proposed in the branch are no longer needed, or if another

solution has been proposed in another branch.

1. Under your repository name, click Pull requests



At the bottom of the pull request, below the comment box,



click Close/Merge pull request.

change in feature #1

Open Saif0872 wants to merge 2 commits into master from new\_feature

Conversation 0 Commits 2 Checks 0 Files changed 8 +10 -1

Saif0872 commented 7 days ago

No description provided.

Saif0872 added 2 commits 7 days ago

- change in feature 90932c0
- Merge branch 'master' of https://github.com/Saif0872/merge\_conflict 1... 6f2c2fa

Add more commits by pushing to the new\_feature branch on Saif0872/merge\_conflict.

Continuous integration has not been set up  
GitHub Actions and several other apps can be used to automatically catch bugs and enforce style.

This branch has no conflicts with the base branch  
Merging can be performed automatically.

Merge pull request You can also open this in GitHub Desktop or view command line instructions.

Reviewers: No reviews. Still in progress? Convert to draft.

Assignees: No one—assign yourself.

Labels: None yet.

Projects: None yet.

Milestone: No milestone.

Development: Successfully merging this pull request may close these issues. None yet.

Write Preview

Leave a comment

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

Close pull request Comment

Remember, contributions to this repository should follow our GitHub Community Guidelines

**3.2 At least One of project member shall create a pull request on a team members repo and close pull requests generated by team members on own Repo as a maintainer.**

Here Pull request created by rajit (my team number)

Code Issues Pull requests 1 Actions Projects Wiki Security Insights Settings

You are screen sharing Stop Share

Label issues and pull requests for new contributors  
Now, GitHub will help potential first-time contributors discover issues labeled with good first issue

Filters is:pr is:open Labels 9 Milestones 0 New pull request

1 Open 0 Closed

Author Label Projects Milestones Reviews Assignee Sort

Update program.cpp  
#1 opened 15 minutes ago by rajit1234



Now I can accept request as maintainer

## Update program.cpp #1

Open rajit1234 wants to merge 1 commit into `feature1` from `feature2`

Conversation 0 Commits 1 Checks 0 Files changed 1

rajit1234 commented 16 minutes ago Collaborator

No description provided.

Update program.cpp V

Add more commits by pushing to the `feature2` branch on `Saif0872/SCM_PROJECT`.

Continuous integration has not been set up  
GitHub Actions and several other apps can be used to automatically catch bugs and enforce style.

This branch has no conflicts with the base branch  
Merging can be performed automatically.

Merge pull request You can also [open this in GitHub Desktop](#) or view [command line instructions](#).

## 3.3 Publish and print network graphs

Aim: Publish and print network graphs

Network Graph:

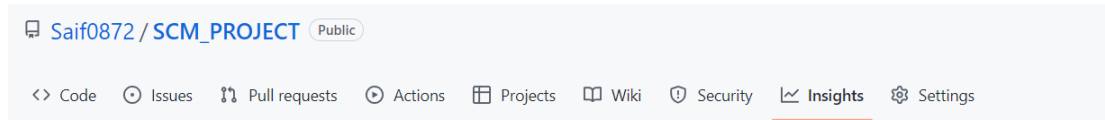
The network graph displays the branch history of the entire repository network, including branches of the root repository and branches of forks that contain commits unique to the network.

### Accessing the network graph

1. On GitHub.com, navigate to the main page of the



2. Under your repository name, click Insights.



2. In the left sidebar, click Network.

