

**Subject Name: Source Code Management**

**Subject Code: CS181**

**Cluster: Beta**

**Department: CSE**

**CHITKARA**  
UNIVERSITY



**Submitted By:**

**Harshita Batra**

**2110990587**

**G8 - B**

## List of Tasks

S No.	Task Title
1.	Setting up of Git Client
2.	Setting up GitHub Account
3.	Generate logs
4.	Create and Visualize branches
5.	Git lifecycle description
6.	Add collaborators on GitHub repository
7.	Fork and commit
8.	Merge and resolve conflicts created due to own activity and collaborators activity.
9.	Reset and revert



# INDEX

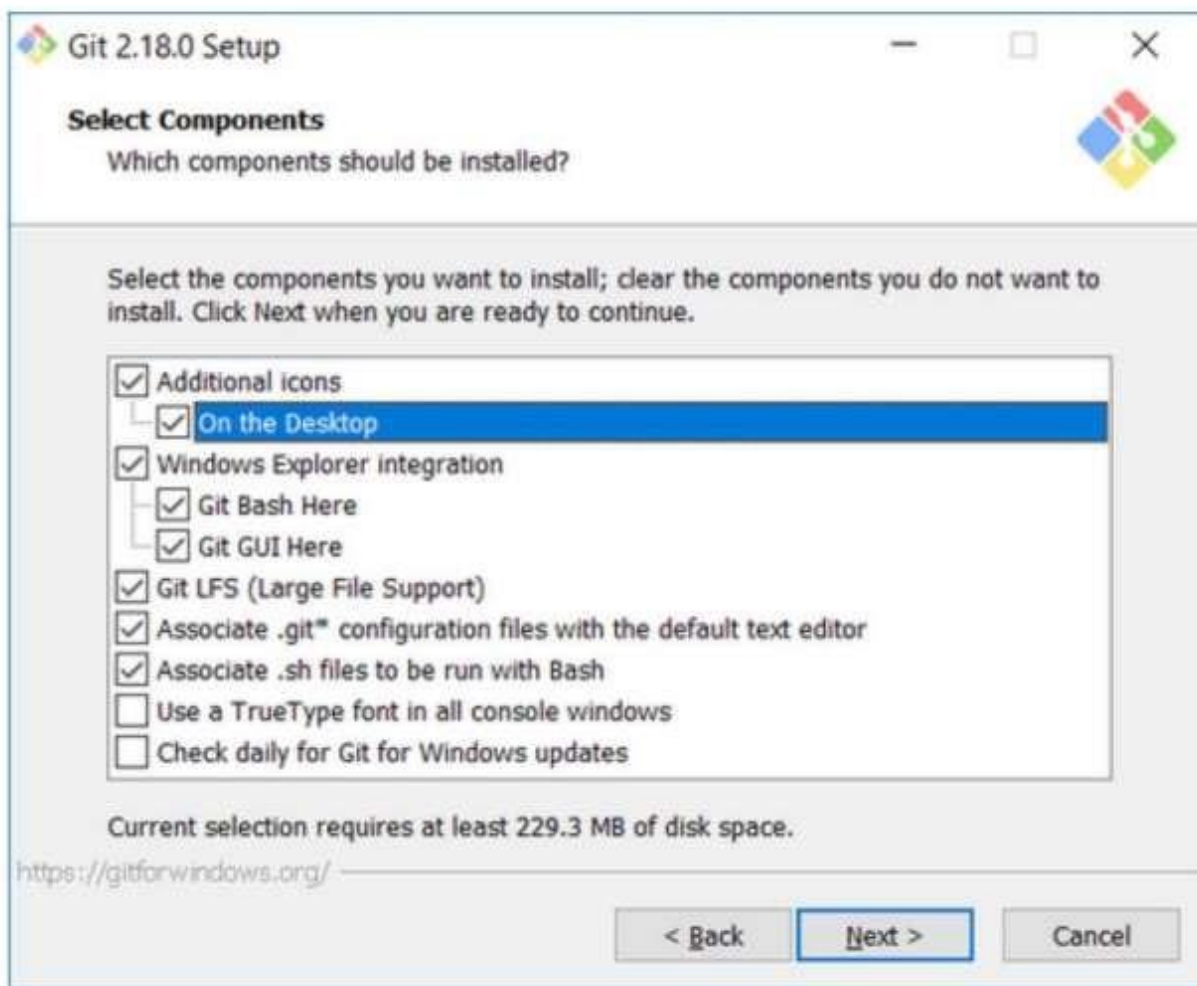
## TASK -1

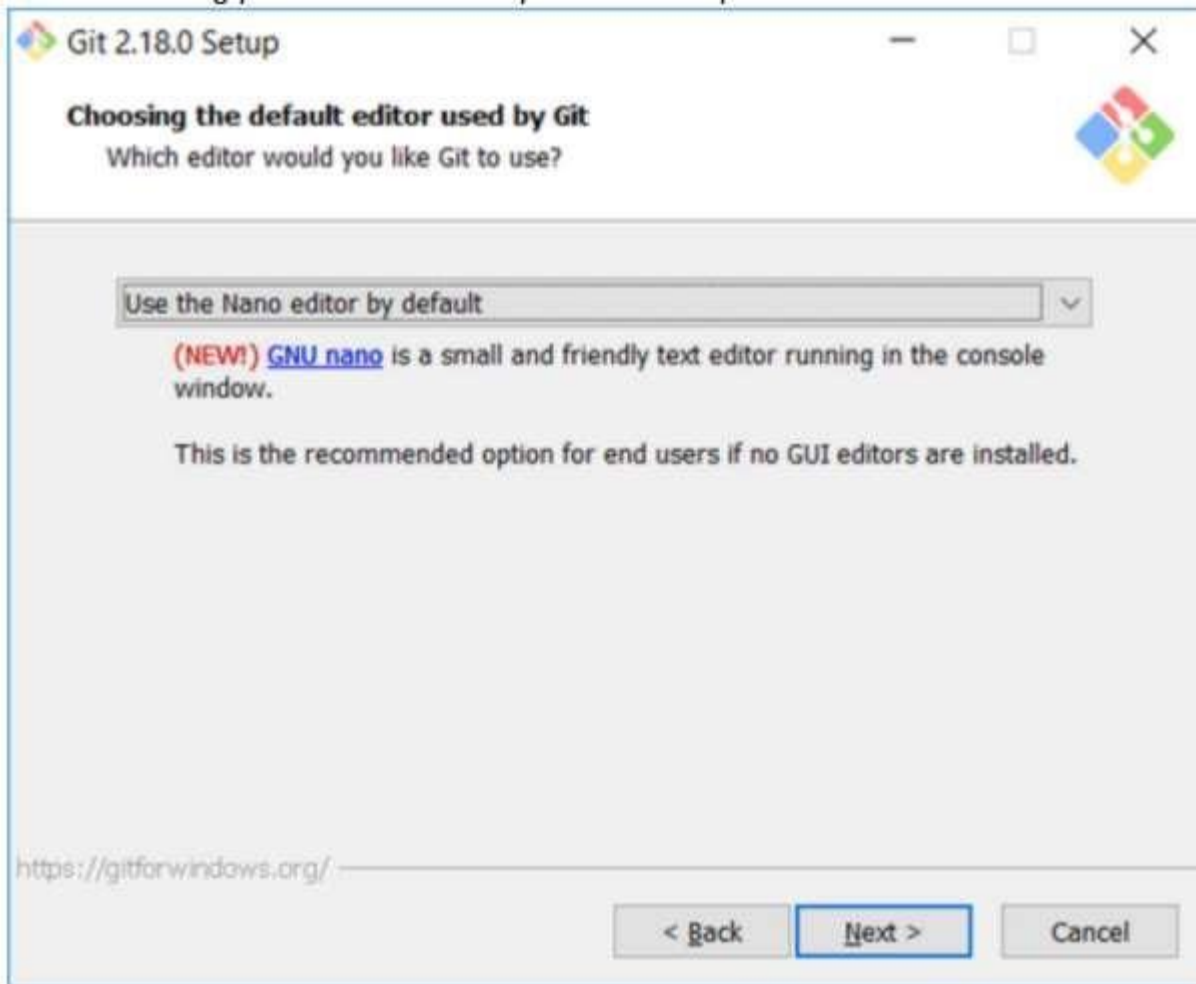
S. No	Aim of Experiment	Remarks by teacher
1	Setting up the git client	
2	Setting up GitHub Account	
3	Program to generate logs	
4	Create and visualize branches in Git	
5	Git life cycle description	

Aim: Setting up the git client.

Git Installation: Download the Git installation program

In the Select Components screen, Windows Explorer Integration is selected as shown:





Three options are acceptable In the Adjusting your PATH screen

1. Use Git from Git Bash only: no integration, and no extra command in your command path.
2. Use Git from the windows Command Prompt: add flexibility – you can simply run git from a windows command prompt, and is often the setting for people in industry – but this does add some extra commands.
3. Use Git and optional Unix tools from the Windows Command Prompt: this is also a robust choice and useful if you like to use Unix like commands like grep



Git 2.18.0 Setup



### Adjusting your PATH environment

How would you like to use Git from the command line?



☒ **Use Git from Git Bash only**

This is the safest choice as your PATH will not be modified at all. You will only be able to use the Git command line tools from Git Bash.

☐ **Use Git from the Windows Command Prompt**

This option is considered safe as it only adds some minimal Git wrappers to your PATH to avoid cluttering your environment with optional Unix tools. You will be able to use Git from both Git Bash and the Windows Command Prompt.

☐ **Use Git and optional Unix tools from the Windows Command Prompt**

Both Git and the optional Unix tools will be added to your PATH.

**Warning:** This will override Windows tools like "find" and "sort". Only use this option if you understand the implications.

<https://gitforwindows.org/>

< Back

Next >

Cancel



Once Git is installed, there is some remaining custom configuration we must do. Follow the steps below:

- a. From within File Explorer, right-click on any folder. A context menu appears containing the commands "Git Bash here" and "GitGUI here". These commands permit you to launch either Git client. For now, select Git Bash here.
- b. Enter the command (replacing name as appropriate) `git config --global core.excludesfile c:/users/name/.gitignore` This tells Git to use the .gitignore file you created in step 2 NOTE: TO avoid typing errors, copy and paste the commands shown here into the Git Bash window, using the arrow keys to edit the red text to match your information.
- c. Enter the command `git config --global user.Email "name@msoe.edu"` This links your Git activity to your email address. Without this, your commits will often show up as "unknown login". Replace name with your own MSOE email name.
- d. Enter the command `git config --global user.name "Your Name"` Git uses this to log your activity. Replace "Your Name" by your actual first and last name. e. Enter the command `git config --global push.default simple` This ensures that all pushes go back to the branch from which they were pulled. Otherwise pushes will go to the master branch, forcing a merge.

### Aim: Setting up the git hub account.

The first steps in starting with GitHub are to create an account, choose a product that fits your needs best, verify your email, set up two-factor authentication, and view your profile.

There are several types of accounts on GitHub. Every person who uses GitHub has their own user account, which can be part of multiple organisations and teams. Your user account is your identity on GitHub.com and represents you as an individual.

1. **Creating an account:** To sign up for an account on GitHub.com, navigate to <https://github.com/> and follow the prompts.

To keep your GitHub account secure you should use a strong and unique password. For more information, see "[Creating a strong password](#)".

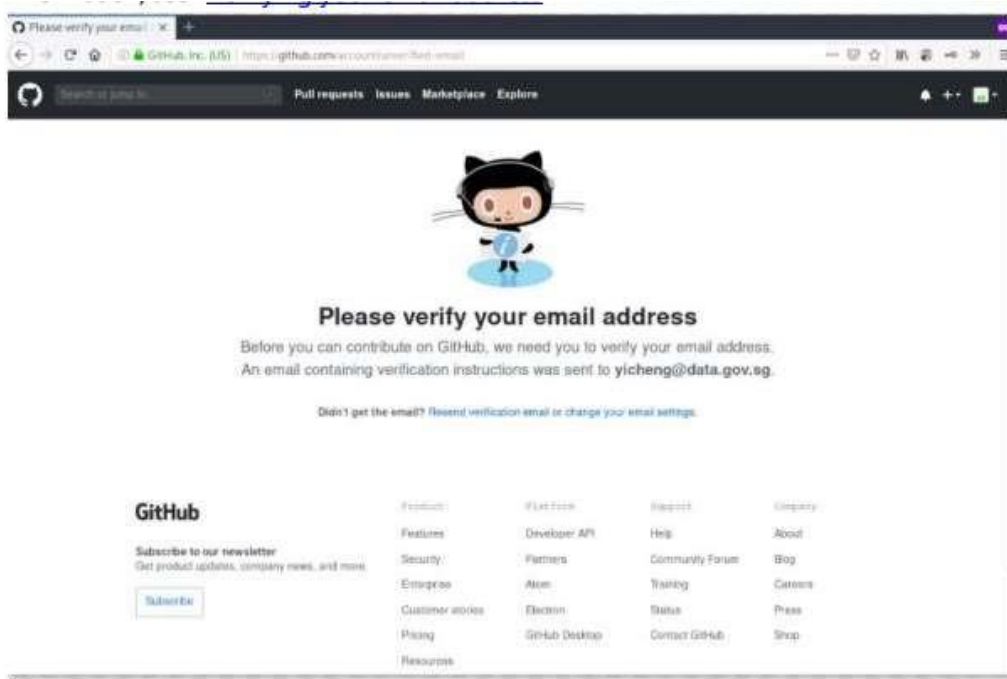


2. **Choosing your GitHub product:** You can choose GitHub Free or GitHub Pro to get access to different

features for your personal account. You can upgrade at any time if you are unsure at first which you want. For more information on all GitHub's plans, see "GitHub's products". product

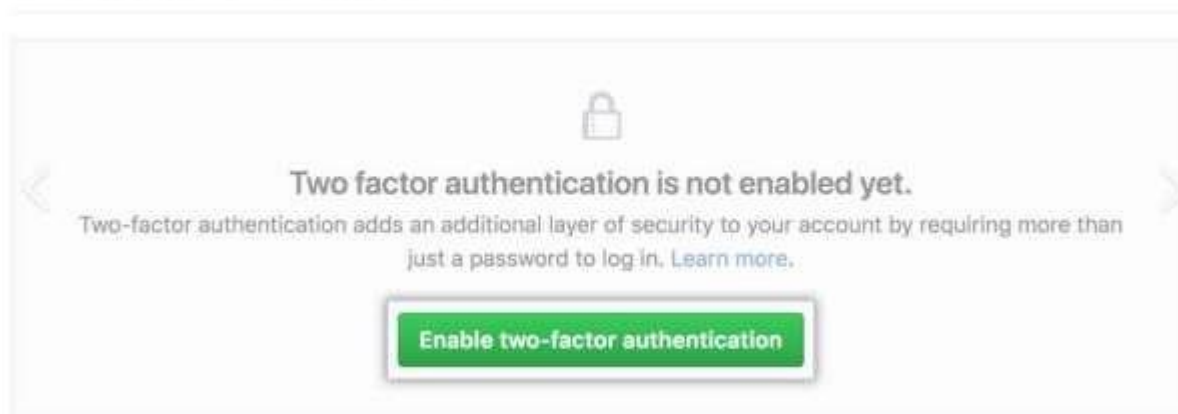


3. Verifying your email address: To ensure you can use all the features in your GitHub plan, verify your email address after signing up for a new account. For more information, see “Verifying your email address.”



4. Configuring two-factor authentication: Two-factor authentication, or 2FA, is an extra layer of security used when logging into websites or apps. We strongly urge you to configure 2FA for safety of your account. For more information, see “About two factor authentication.”

## Two-factor authentication



5. Viewing your GitHub profile and contribution graph: Your GitHub profile tells people the story of your work through the repositories and gists you’ve pinned, the organization memberships you’ve chosen to publicize, the contributions you’ve made, and the projects you’ve created. For more information, see “About your profile” and “Viewing contributions on your profile.”

11 contributions in the last year

Contribution settings ▾



## Aim: Program to generate logs

### Basic Git init

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

### Basic Git status

The git status command displays the state of the working directory and the staging area. It lets you see which changes have been staged, which haven't, and which files aren't being tracked by Git. Status output does not show you any information regarding the committed project history.

### Basic Git commit

The git commit command captures a snapshot of the project's currently staged changes. Committed snapshots can be thought of as “safe” versions of a project—Git will never change them unless you explicitly ask it to. Prior to the execution of git commit, The git add command is used to promote or 'stage' changes to the project that will be stored in a commit. These two commands git commit and git add are two of the most frequently used

### Basic Git add command

The git add command adds a change in the working directory to the staging area. It tells Git that you want to include updates to a particular file in the next commit. However, git add doesn't really affect the repository in any significant way—changes are not actually recorded until you run git commit. Basic Git log

Git log command is one of the most usual commands of git. It is the most useful command for Git. Every time you need to check the history, you have to use the git log command. The basic git log command will display the most recent commits and the status of the head. It will use as:

MINGW64:/h/Projects/Code blocks project

```
BATRA's@DESKTOP-I2LH7RN MINGW64 /h/Projects/Code blocks project
$ git config --global user.name
Harshita-Batra-1

BATRA's@DESKTOP-I2LH7RN MINGW64 /h/Projects/Code blocks project
$ git config --global user.email
harshita0587.be21@chitkara.edu.in

BATRA's@DESKTOP-I2LH7RN MINGW64 /h/Projects/Code blocks project
$ git status
fatal: not a git repository (or any of the parent directories): .git

BATRA's@DESKTOP-I2LH7RN MINGW64 /h/Projects/Code blocks project
$ git init
Initialized empty Git repository in H:/Projects/Code blocks project/.git/

BATRA's@DESKTOP-I2LH7RN MINGW64 /h/Projects/Code blocks project (master)
$ git add -A
```

```
$ git status
On branch master

No commits yet

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

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

```
BATRA's@DESKTOP-I2LH7RN MINGW64 /h/Projects/Code blocks project (master)
$ git commit -m"Codes"
[master (root-commit) beac36f] Codes
73 files changed, 1275 insertions(+)
```

```
BATRA's@DESKTOP-I2LH7RN MINGW64 /h/Projects/Code blocks project (master)
$ git log
commit beac36f4e5e2b26bdf8466cea172b7b606db79e0 (HEAD -> master)
Author: Harshita-Batra-1 <harshita0587.be21@chitkara.edu.in>
Date: Sun Apr 10 13:16:49 2022 +0530
```

#### Codes

```
BATRA's@DESKTOP-I2LH7RN MINGW64 /h/Projects/Code blocks project (master)
$ git remote add origin https://github.com/Group08-Chitkara-University/2110990587.git
git push -u origin main
BATRA's@DESKTOP-I2LH7RN MINGW64 /h/Projects/Code blocks project (master)
$ git branch -M main
```

```
BATRA's@DESKTOP-I2LH7RN MINGW64 /h/Projects/Code blocks project (main)
$ git push -u origin main
Enumerating objects: 134, done.
Counting objects: 100% (134/134), done.
Delta compression using up to 4 threads
Compressing objects: 100% (90/90), done.
Writing objects: 100% (134/134), 215.20 KiB | 2.95 MiB/s, done.
Total 134 (delta 39), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (39/39), done.
To https://github.com/Group08-Chitkara-University/2110990587.git
 * [new branch]      main -> main
Branch 'main' set up to track remote branch 'main' from 'origin'.
```

## Aim: Create and visualize branches in Git

### How to create branches?

The main branch in git is called master branch. But we can make branches out of this main master branch. All the files present in master can be shown in branch but the files which are created in branch are not shown in master branch. We also can merge both the parent(master) and child (other branches).

1. For creating a new branch: `git branch "name of branch"`
2. To check how many branches we have : `git branch`
3. To change the present working branch: `git checkout "name of the branch"`

### Visualizing Branches:

To visualize, we have to create a new file in the new branch "activity1" instead of the master branch. After this we have to do three step architecture i.e. working directory, staging area and git repository.

After this I have done the 3 Step architecture which is tracking the file, send it to staging area and finally we can rollback to any previously saved version of this file.

After this we will change the branch from activity1 to master, but when we switch back to master branch the file we created i.e "hello" will not be there. Hence the new file will not be shown in the master branch. In this way we can create and change different branches. We can also merge the branches by using the `git merge` command.

In this way we can create and change different branches. We can also merge the branches by using `git merge` command.

```
MINGW64:/h/Projects/Code blocks project/1HelloWord

BATRA's@DESKTOP-I2LH7RN MINGW64 /h/Projects/Code blocks project/1HelloWord (main)
$ ls
HelloWord.cbp  HelloWord.layout  bin/  main.cpp  obj/

BATRA's@DESKTOP-I2LH7RN MINGW64 /h/Projects/Code blocks project/1HelloWord (main)
$ git branch
  feature
* main

BATRA's@DESKTOP-I2LH7RN MINGW64 /h/Projects/Code blocks project/1HelloWord (main)
$ git checkout feature
Switched to branch 'feature'
```



```

BATRA's@DESKTOP-I2LH7RN MINGW64 /h/Projects/Code blocks project/1HelloWord (feature)
$ git branch
* feature
  main

BATRA's@DESKTOP-I2LH7RN MINGW64 /h/Projects/Code blocks project/1HelloWord (feature)
$ ls
HelloWord.cbp  HelloWord.layout  bin/  main.cpp  obj/

BATRA's@DESKTOP-I2LH7RN MINGW64 /h/Projects/Code blocks project/1HelloWord (feature)
$ ls -ah
./  ../  .main.cpp.swp  HelloWord.cbp  HelloWord.layout  bin/  main.cpp  obj/

BATRA's@DESKTOP-I2LH7RN MINGW64 /h/Projects/Code blocks project/1HelloWord (feature)
$ vi main.cpp

BATRA's@DESKTOP-I2LH7RN MINGW64 /h/Projects/Code blocks project/1HelloWord (feature)
$ git status
On branch feature
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:   main.cpp

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

BATRA's@DESKTOP-I2LH7RN MINGW64 /h/Projects/Code blocks project/1HelloWord (feature)
$ git add -A

BATRA's@DESKTOP-I2LH7RN MINGW64 /h/Projects/Code blocks project/1HelloWord (feature)
$ git commit -m"Changing file in feature branch"
[feature 2a6fe76] Changing file in feature branch

```

🔗 MINGW64:/h/Projects/Code blocks project/1HelloWord

1 file changed, 3 insertions(+), 4 deletions(-)

```

BATRA's@DESKTOP-I2LH7RN MINGW64 /h/Projects/Code blocks project/1HelloWord (feature)
$ log --oneline
bash: log: command not found

```

```

BATRA's@DESKTOP-I2LH7RN MINGW64 /h/Projects/Code blocks project/1HelloWord (feature)
$ git log --oneline
2a6fe76 (HEAD -> feature) Changing file in feature branch
3e1ecf0 intial commit
1715af2 Added code in previous file
beac36f (origin/main, main) Codes

```

```

BATRA's@DESKTOP-I2LH7RN MINGW64 /h/Projects/Code blocks project/1HelloWord (feature)
$ cat main.cpp
#include <iostream>

```

```
using namespace std;
```

```

int main()
{
    cout << "Hello world!" << endl; //simple program
    cout << "Hello's" << endl;
    cout << "Hello\"s coder\"s " << endl; //to add " in output as well we use \
    cout << "'Hello'" << endl;
    cout << "\"Hello\"" << endl; //this is wrong only doubles are used(in python-single can be)
    cout << "\"Hello world\"" << endl;
    cout << "Hello\"s what's up!" << endl;
    return 0;
};wq

```

```

BATRA's@DESKTOP-I2LH7RN MINGW64 /h/Projects/Code blocks project/1HelloWord (feature)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

```

```

BATRA's@DESKTOP-I2LH7RN MINGW64 /h/Projects/Code blocks project/1HelloWord (main)
$ cat main.cpp
#include <iostream>

```

```
using namespace std;
```

```

int main()
{
    cout << "Hello world!" << endl;
    cout << "Hello's" << endl;
    cout << "Hello\"s coder\"s " << endl;
    cout << "'Hello'" << endl;
    cout << "\"Hello\"" << endl; //this is wrong only doubles are used(in python-single can be)
    cout << "\"Hello world\"" << endl;

```

```
    cout << "\\Hello world\\"<< endl;  
    return 0;  
}
```

```
BATRA's@DESKTOP-I2LH7RN MINGW64 /h/Projects/Code blocks project/1HelloWord (main)  
$ git log --oneline  
beac36f (HEAD -> main, origin/main) Codes
```

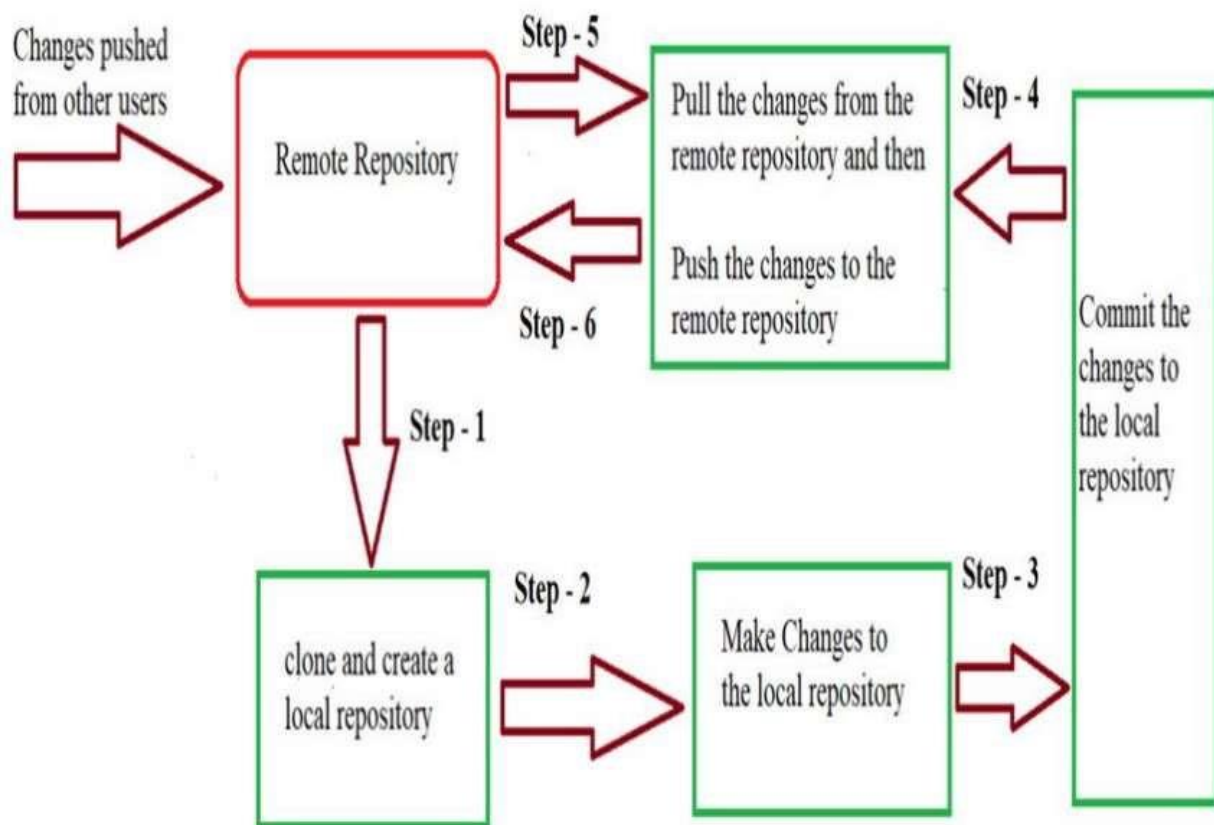
```
BATRA's@DESKTOP-I2LH7RN MINGW64 /h/Projects/Code blocks project/1HelloWord (main)  
$ git checkout feature  
Switched to branch 'feature'
```

```
BATRA's@DESKTOP-I2LH7RN MINGW64 /h/Projects/Code blocks project/1HelloWord (feature)  
$ git log --oneline  
2a6fe76 (HEAD -> feature) Changing file in feature branch  
3e1ecf0 initial commit  
1715af2 Added code in previous file  
beac36f (origin/main, main) Codes
```



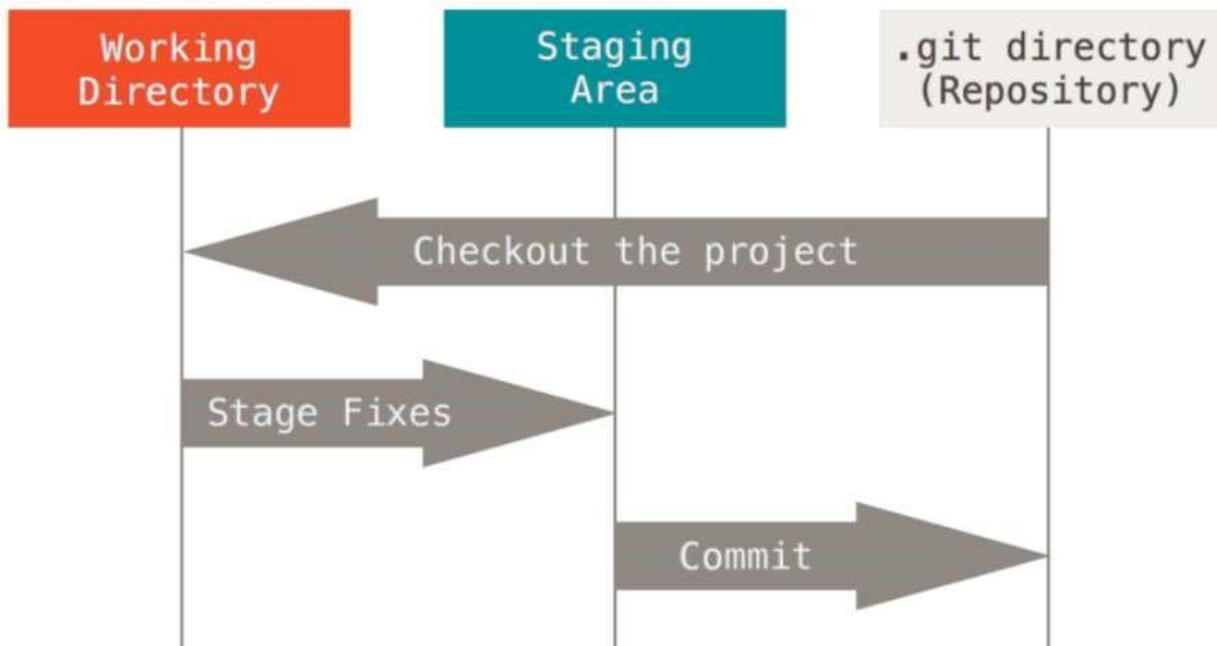
## Aim: Git Lifecycle description

Git is used in our day-to-day work, we use Git for keeping a track of our files, working in a collaboration with our team, to go back to our previous code versions if we face some error. Git helps us in many ways. Let us look at the Lifecycle description that git has and understand more about its life cycle. Let us see some of the basic steps that we have to follow while working with Git-



- **Step 1-** We first clone any of the code residing in the remote repository to make our own local repository.
- **Step 2-** We edit the files that we have cloned in our local repository and make the necessary changes in it.
- **Step 3-** We commit our changes by first adding them to our staging area and committing them with a commit message.
- **Step 4 and Step 5-** We first check whether there are any of the changes done in the remote repository by some other users and we first pull that changes.
- **Step 6-** If there are no changes we push our changes to the remote repository and we are done with our work.

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



## 1. Working Directory

Whenever we want to initialize our local project directory to make a Git repository, we use the `git init` command. After this command, git becomes aware of the files in the project although it does not track the files yet. The files are further tracked in the staging area.

## 2. Staging Area

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

`git add<filename>`

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

# TASK 1.2

S. No.	Title	Remarks
6	Add collaborators on GitHub Repo	
7	Fork and Commit	
8	Merge and resolve conflicts created due to own activity and collaborators activity	
9	Reset and revert	

**Aim:** Add collaborators on GitHub Repository.

**Theory:**

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.
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 **on Collaborators & teams**.
5. Click on **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. In the search field, start typing the name of person you want to invite, then click a name in the list of matches.

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.

```
BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/Clonning (master)
$ git clone https://github.com/Harshita-Batra-1/0242_anushka.git
Cloning into '0242_anushka'...
remote: Enumerating objects: 14, done.
remote: Counting objects: 100% (14/14), done.
remote: Compressing objects: 100% (13/13), done.
remote: Total 14 (delta 4), reused 3 (delta 0), pack-reused 0
Receiving objects: 100% (14/14), done.
Resolving deltas: 100% (4/4), done.

BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/Clonning (master)
$ git remote add upstream https://github.com/Harshita-Batra-1/0242_anushka.git

BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/Clonning (master)
$ git remote -v
upstream      https://github.com/Harshita-Batra-1/0242_anushka.git (fetch)
upstream      https://github.com/Harshita-Batra-1/0242_anushka.git (push)
```

← → ▾ ↑ 📁 > This PC > Desktop > Clonning			
^			
Name	Date modified	Type	Size
📁 0242_anushka	5/20/2022 5:14 PM	File folder	

The Collaborator can now make a change in her clone of the Owner's repository, exactly the same way as we've been doing before:

```
$ nano pluto.txt
$ cat pluto.txt

It is so a planet!

$ git add pluto.txt
$ git commit -m "Add notes about Pluto"

1 file changed, 1 insertion(+)
create mode 100644 pluto.txt
```

Then push the change to the *Owner's repository* on GitHub:

```
$ git push origin main

Enumerating objects: 4, done.
Counting objects: 4, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 306 bytes, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/vlad/planets.git
 9272da5..29aba7c  main -> main
```

Note that we didn't have to create a remote called `origin`: Git uses this name by default when we clone a repository. (This is why `origin` was a sensible choice earlier when we were setting up remotes by hand.)

Take a look at the Owner's repository on GitHub again, and you should be able to see the new commit made by the Collaborator. You may need to refresh your browser to see the new commit.

Local repository has had a single "remote", called `origin`. Remote is copy of the repository that is hosted.



- `git remote -v` lists all the remotes that are configured (we already used this in the last episode)
- `git remote add [name] [url]` is used to add a new remote
- `git remote remove [name]` removes a remote. Note that it doesn't affect the remote repository at all - it just removes the link to it from the local repo.
- `git remote set-url [name] [newurl]` changes the URL that is associated with the remote. This is useful if it has moved, e.g. to a different GitHub account, or from GitHub to a different hosting service. Or, if we made a typo when adding it!
- `git remote rename [oldname] [newname]` changes the local alias by which a remote is known - its name. For example, one could use this to change upstream to fred.

To download the Collaborator's changes from GitHub, the Owner now enters:

```
$ git pull origin main
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 3 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
From https://github.com/vlad/planets
* branch          main      -> FETCH_HEAD
  9272da5..29aba7c main      -> origin/main
Updating 9272da5..29aba7c
Fast-forward
 pluto.txt | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 pluto.txt
```

Now all 3 repositories are in sync (Owner's local, Collaborator's local, Owner's on git hub)



## **Aim: Fork and Commit**

**A fork is a copy of a repository that we manage.**

**Forks let us make changes to a project without affecting the original repository. We can fetch updates from or submit changes to the original repository with pull requests.**

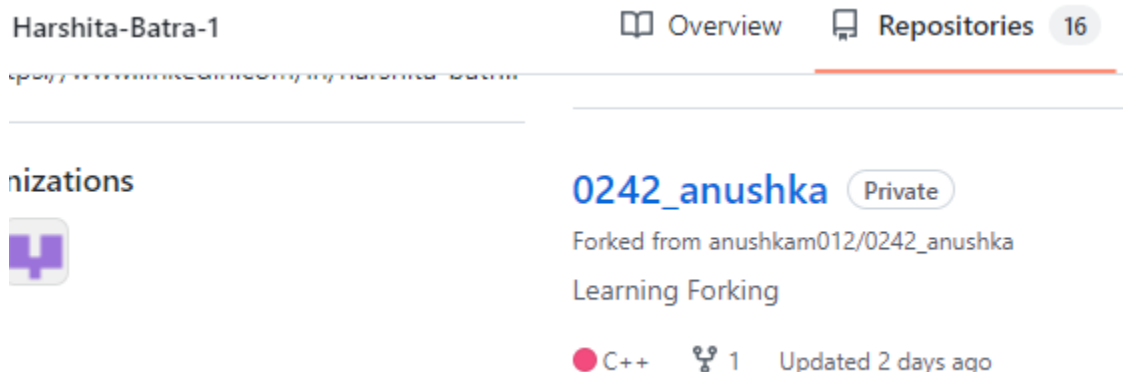
## **Theory:**

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 'Smart Parking System', 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.

## **FORKED FROM TEAM MEMBER'S REPOSITORY -**



The screenshot shows the GitHub profile of 'Harshita-Batra-1' with tabs for 'Overview' and 'Repositories' (16). Below the profile, the 'Organizations' section is partially visible. The main content area displays a repository named '0242\_anushka' with a 'Private' label. It indicates the repository was 'Forked from anushkam012/0242\_anushka' and is titled 'Learning Forking'. The repository is in the 'C++' language, has 1 star, and was updated 2 days ago.

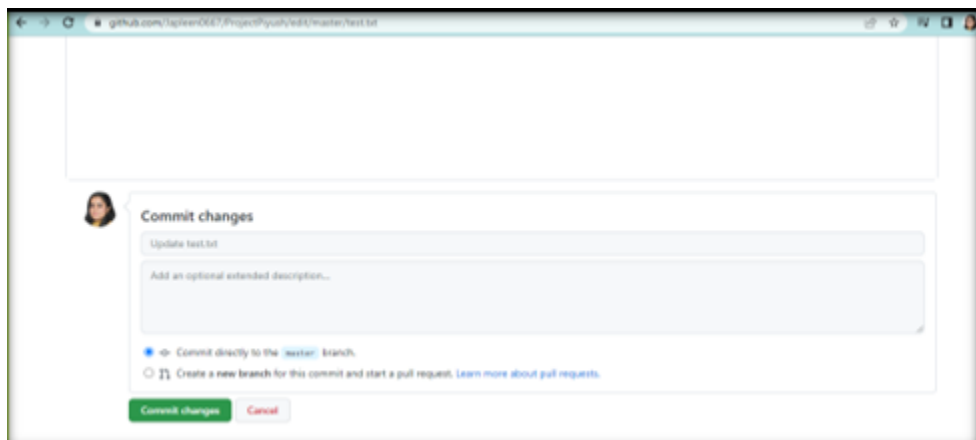
After forking forked repository copied in your account now you can commit changes. Here is how:

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

Note: You need repository access to commit changes otherwise you have to do pull request.

4. Do Modification and click on commit changes you can also commit in new branch by selecting create a new branch.
5. Then, click on commit change now, Fork commit done!

## Committed changes in forked repository –



Forked!

**Flutter-Plants-Identification** Public

Forked from brakenseddik/Flutter-Plants-Identification

Forked! (A mobile application written with flutter & firebase & Tensorflow <3  
for plants identification (ML))

Dart 17 Updated on Oct 21, 2020

Star

## **Aim - Merge and Resolve conflicts created due to own Activity and Collaborators activity**

**There are a few steps that could reduce the steps needed to resolve merge conflicts in Git.**

- 1. The easiest way to resolve a conflicted file is to open it and make any necessary changes.**
- 2. After editing the file, we can use the git add a command to stage the new merged content.**
- 3. The final step is to create a new commit with the help of the git commit command.**
- 4. Git will create a new merge commit to finalize the merge**

**Let us now look into the Git commands that may play a significant role in resolving conflicts.**

MINGW64:/c/Users/Harshita batra/Desktop/Code

```
BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/Code (master)
$ git init
Initialized empty Git repository in C:/Users/Harshita batra/Desktop/Code/.git/

BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/Code (master)
$ git add .

BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/Code (master)
$ git commit -m"Initial"
[master (root-commit) 08acca7] Initial
 1 file changed, 14 insertions(+)
 create mode 100644 main.cpp

BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/Code (master)
$ git remote add origin https://github.com/Harshita-Batra-1/Code-Merge.git
git push -u origin main
BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/Code (master)
$ git branch -M main

BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/Code (main)
$ git push -u origin main
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 397 bytes | 397.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/Harshita-Batra-1/Code-Merge.git
 * [new branch]      main -> main
Branch 'main' set up to track remote branch 'main' from 'origin'.
```

14 lines (12 sloc) | 345 Bytes

```
1  #include <iostream>
2
3  using namespace std;
4
5  int main()
6  {
7      cout << "Hello world!" << endl;
8      cout << "Hello's" << endl;
9      cout << "Hello\"s coder\"s " << endl;
10     cout << "'Hello'" << endl;
11     cout << "\"Hello\"" << endl; //this is wrong only doubles are used(in python-single can be)
12     cout << "\"Hello world\"" << endl;
13     return 0;
14 }
```

MINGW64:/c/Users/Harshita batra/Desktop/Code

```
#include <iostream>

using namespace std;

int main()
{
    cout << "Hello world!" << endl;
    |
    cout << "Hello\"s coder\"s " << endl;
    cout << "'Hello'" << endl;
    cout << "\"Hello\"" << endl; //this is wrong only doubles are used
be)
    cout << "\"Hello world\"" << endl;
    return 0;
}
~
```

BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/Code (main)

\$ vi main.cpp

BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/Code (main)

\$

BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/Code (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: main.cpp

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

BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/Code (main)

\$ git add .

BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/Code (main)

\$ git commit -m"Second change"

[main fc1d99d] Second change

1 file changed, 1 insertion(+), 1 deletion(-)

MINGW64:/c/Users/Harshita batra/Desktop/Code

```
#include <iostream>

using namespace std;

int main()
{
    cout << "Hello world!" << endl;
    cout << "Hello's" << endl;|
    cout << "Hello\"s coder\"s " << endl;
    cout << "'Hello'" << endl;
    cout << "\\Hello\\" << endl; //this is wrong only doubles are use
be)
    cout << "\"Hello world\"" << endl;
    return 0;
}
~
~
~
~
~
~
~
~
~
~
main.cpp[+] [dos] (00:25 22/05/2022)
-- INSERT --
```

```
BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/Code (main)
$ git branch dev

BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/Code (main)
$ git checkout dev
Switched to branch 'dev'

BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/Code (dev)
$ vi main.cpp

BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/Code (dev)
$ git status
On branch dev
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:   main.cpp

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

BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/Code (dev)
$ git add .

BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/Code (dev)
$ git commit -m"Committed in dev branch"
[dev d5d7d09] Committed in dev branch
1 file changed, 1 insertion(+), 1 deletion(-)
```

```
BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/Code (dev)
$ vi main.cpp

BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/Code (dev)
$ git status
On branch dev
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:   main.cpp

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

BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/Code (dev)
$ git add .

BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/Code (dev)
$

BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/Code (dev)
$ git commit -m"Changes in dev branch"
[dev cc4c297] Changes in dev branch
1 file changed, 4 insertions(+)

BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/Code (dev)
$ git checkout main
Switched to branch 'main'
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/Code (main)
$ git log
commit fc1d99d8fa0775fc5f196a1241ae650c710faa2d (HEAD -> main)
Author: Harshita-Batra-1 <harshita0587.be21@chitkara.edu.in>
Date:   Sun May 22 00:26:09 2022 +0530

    Second change

commit 08acca7893338faa85d603e6f07e4f20f847bc47 (origin/main)
Author: Harshita-Batra-1 <harshita0587.be21@chitkara.edu.in>
Date:   Sun May 22 00:14:36 2022 +0530

    Initial
```

dev ▾

Code-Merge / main.cpp



Harshita-Batra-1 Committed in dev branch

1 contributor

14 lines (12 sloc) | 345 Bytes

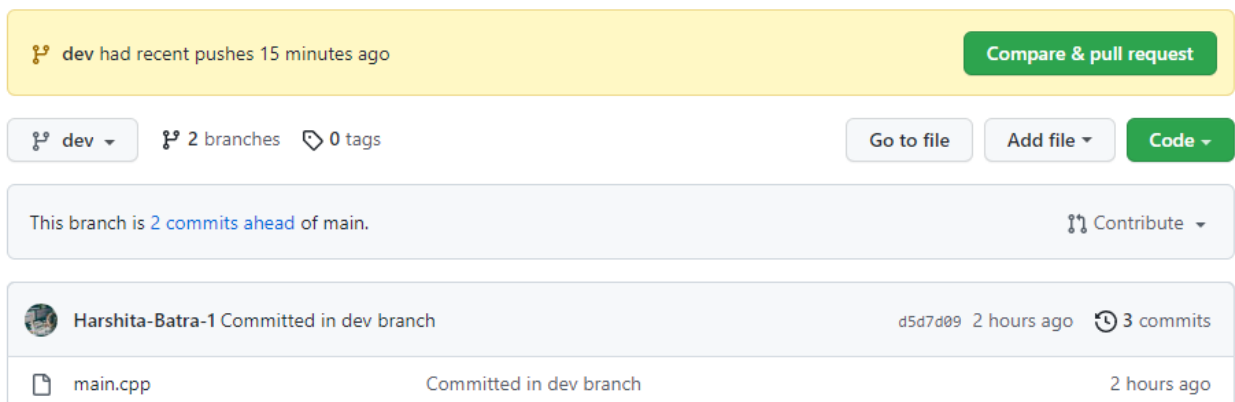
```
1  #include <iostream>
2
3  using namespace std;
4
5  int main()
6  {
7      cout << "Hello world!" << endl;
8      cout << "Hello's" << endl;
9      cout << "Hello\"s coder\"s " << endl;
10     cout << "'Hello'" << endl;
11     cout << "\"Hello\"" << endl; //this is wrong only doubles are used(in python-single can be)
12     cout << "\"Hello world\"" << endl;
13     return 0;
14 }
```

## STEPS-

- To open a pull request we first have to make a new branch, by using git branch *branch name* option.
- After making new branch we add a file to the branch or make changes in the existing file.
- Add and commit the changes to the local repository.
- Use git push origin *branchname* option to push the new branch to the main repository.
- After pushing new branch Github will either automatically ask you to create a pull request or you can create your own pull request.



- To create your own pull request click on pull request option.
- Github will detect any conflicts and ask you to enter a description of your pull request.
- After opening a pull request all the team members will be sent the request if they want to merge or close the request.
- If the team member chooses not to merge your pull request they will close you're the pull request.
- To close the pull request simply click on close pull request and add comment/ reason why you closed the pull request.
- You can see all the pull request generated and how they were dealt with by clicking on pull request option and below are the images.



# 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 branches](#)



base: main ▾



compare: dev ▾

✓ **Able to merge.** These branches can be automatically merged.

Discuss and review the changes in this comparison with others. [Learn about pull requests](#)

2 commits

0 files changed

Harshita-Batra-1 / Code-Merge Public

Pin



<> Code Issues Pull requests 1 Actions Projects Wiki Security Insights Settings

## Dev #1



Open Harshita-Batra-1 wants to merge 2 commits into main from dev

Conversation 0

Commits 2

Checks 0

Files changed 0



Harshita-Batra-1 commented now

Owner



Trying pull request across 2 branches



Harshita-Batra-1 added 2 commits 2 hours ago



Second change

fc1d99d



Committed in dev branch

d5d7d09

# Dev #1

**Merged** Harshita-Batra-1 merged 2 commits into `main` from `dev` 14 seconds ago

Conversation 0 Commits 2 Checks 0 Files changed 0



Harshita-Batra-1 commented 1 minute ago

Owner ...

Trying pull request across 2 branches



Harshita-Batra-1 added 2 commits 2 hours ago



Second change

fc1d99d



Committed in dev branch

d5d7d09



Harshita-Batra-1 merged commit 2eeffe8 into `main` 14 seconds ago

Revert



**Pull request successfully merged and closed**

Delete branch

You're all set—the `dev` branch can be safely deleted.

<> Code Issues Pull requests Actions Projects Wiki Security Insights Settings

main 1 branch 0 tags

Go to file

Add file

Code



Harshita-Batra-1 Merge pull request #1 from Harshita-Batra-1/dev

2eeffe8 2 minutes ago 4 commits



main.cpp

Initial

2 hours ago

Harshita-Batra-1 / Learning-Branch-Merge Public

Pin

Unwatch 1

Fork 0

Star 0

<> Code Issues Pull requests Actions Projects Wiki Security Insights Settings

main

Commits on May 22, 2022

Merge pull request #1 from Harshita-Batra-1/dev

Verified



2eeffe8



Harshita-Batra-1 committed 4 minutes ago

Committed in dev branch



d5d7d09



Harshita-Batra-1 committed 2 hours ago

Second change



fc1d99d



Harshita-Batra-1 committed 2 hours ago

Initial



00acca7



Harshita-Batra-1 committed 2 hours ago

## Case of Conflict merging due to own activity-

### Theory:

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

1. 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”.

2. Then we will make a new branch and so modification in one very same line.
3. Again, do add commit and push.
4. Now go back to main branch using **git checkout main**.
5. Again, do change in same line.
6. Then add commit and push in main branch.
7. Switch to new dev and do pull request using ‘**git pull origin main**’ and then you can see conflict error.

```
BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/Code (main)
$ git branch
dev
* main
```

```
BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/Code (main)
$ cd Branch

BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/Code/Branch (main)
$ touch file.txt

BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/Code/Branch (main)
$ git checkout dev
Switched to branch 'dev'

BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/Code/Branch (dev)
$ ls
file.txt

BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/Code/Branch (dev)
$ cat file.txt

BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/Code/Branch (dev)
$ cat file.txt
Hello
I use git and github
BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/Code/Branch (dev)
$ git add .

BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/Code/Branch (dev)
$ git commit -m"initial"
[dev 5bf21f3] initial
1 file changed, 2 insertions(+)
create mode 100644 Branch/file.txt
```

```

BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/Code/Branch (dev)
$ git checkout main
Deletion of directory 'Branch' failed. Should I try again? (y/n) y
Deletion of directory 'Branch' failed. Should I try again? (y/n) n
Switched to branch 'main'
Your branch is ahead of 'origin/main' by 4 commits.
  (use "git push" to publish your local commits)

BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/Code/Branch (main)
$ vim file.txt

BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/Code/Branch (main)
$ git add .
warning: LF will be replaced by CRLF in Branch/file.txt.
The file will have its original line endings in your working directory

BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/Code/Branch (main)
$ git commit -m"Initialcommit in main branch"
[main cbe0f92] Initialcommit in main branch
 1 file changed, 1 insertion(+)
 create mode 100644 Branch/file.txt

BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/Code/Branch (main)
$ git checkout dev
Switched to branch 'dev'

BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/Code/Branch (dev)
$ cat file.txt
Hello
I use git and github
BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/Code/Branch (dev)
$ git checkout main
Switched to branch 'main'
Your branch is ahead of 'origin/main' by 5 commits.
  (use "git push" to publish your local commits)

BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/Code/Branch (main)
$ cat file.txt

BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/Code/Branch (main)
$ git checkout dev
Switched to branch 'dev'

BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/Code/Branch (dev)
$ git merge main
Auto-merging Branch/file.txt
CONFLICT (add/add): Merge conflict in Branch/file.txt
Automatic merge failed; fix conflicts and then commit the result.

BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/Code/Branch (dev|MERGING)
$ git mergetool

```

Switch back to dev branch because conflict is in new feature branch using **git checkout dev**.

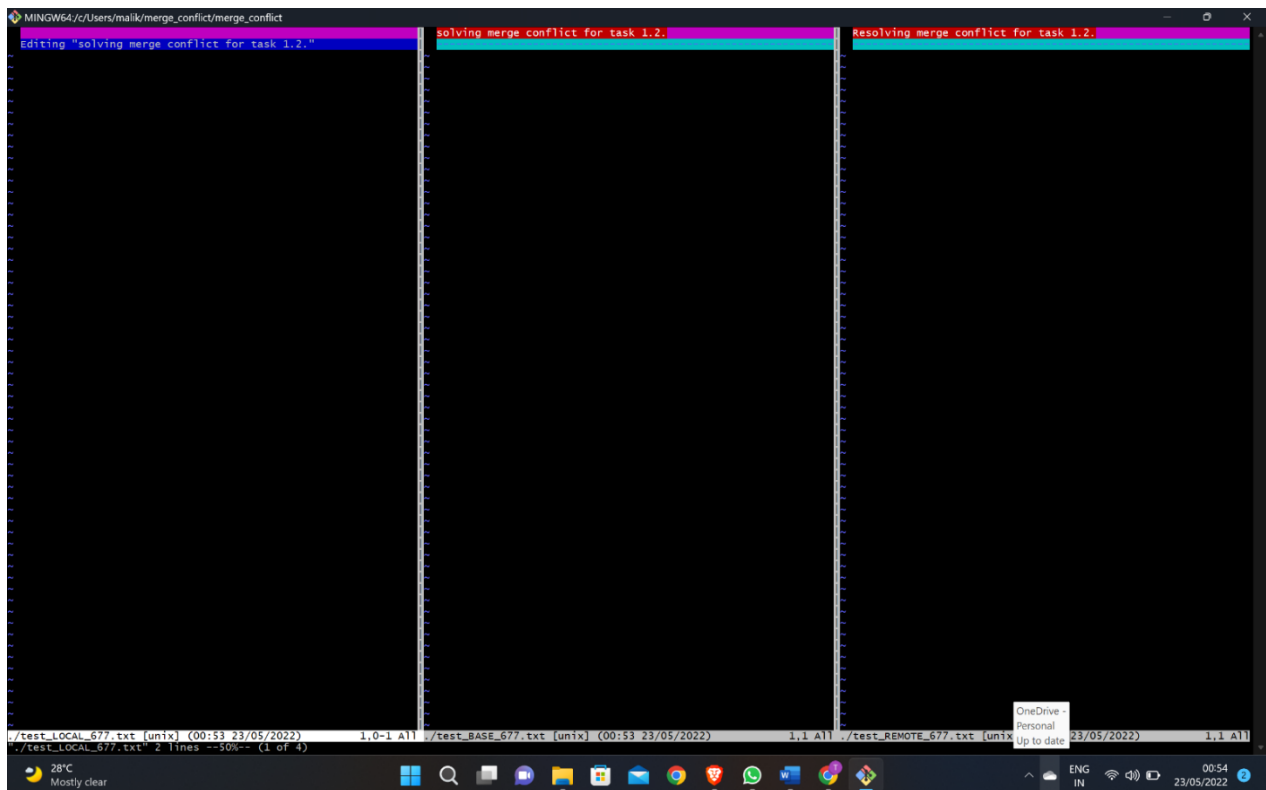
Now there are three ways to solve conflict:

- Using GitHub.
- Using normal editor.
- And we can use mergetool via git.

Here we are going to solve using mergetool via git.

### 1. Type **mergetool**.

After mergetool successfully configured, A pop open as you can see below.



1. Now decide which modification you want to keep.
2. Press 'i' to remove unwanted character and edit and press esc (for exiting insert mode).
3. Then type ': wq' for exiting mergetool.
4. After exiting, do add and commit and push.

Note: Keep in mind we don't have to write commit message because it automatically adds a commit message.

```

BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/Code/Branch (dev|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 diffmerge ecmerge p4merge araxis bc codecompare
smmerge emerge vimdiff nvimdiff
Merging:
Branch/file.txt

Normal merge conflict for 'Branch/file.txt':
  {local}: created file
  {remote}: created file
Hit return to start merge resolution tool (vimdiff):
3 files to edit

BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/Code/Branch (dev|MERGING)
$ git commit -m"Merged after resolving conflicts"
[dev a23ecc6] Merged after resolving conflicts

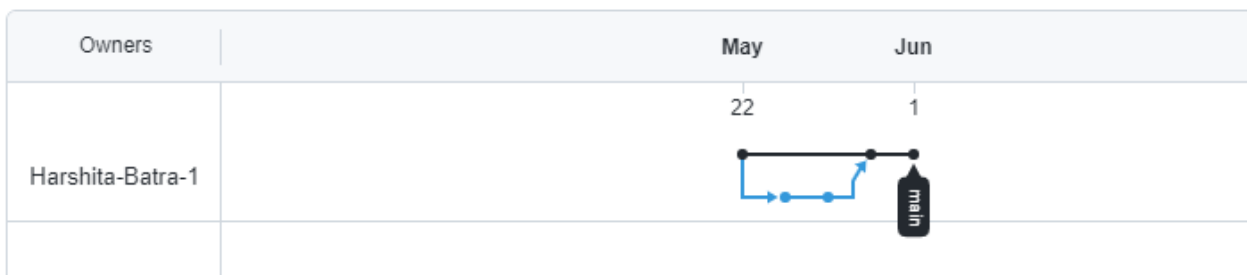
BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/Code/Branch (dev)
$ git log --oneline
a23ecc6 (HEAD -> dev) Merged after resolving conflicts
cbe0f92 (main) Initialcommit in main branch
5bf21f3 initial
cc4c297 Changes in dev branch
86854c2 Committed in dev branch hjsx abdchj kqncuilk
d5d7d09 (origin/dev) Committed in dev branch
fc1d99d Second change
08acca7 (origin/main) Initial

BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/Code/Branch (dev)
$ git log --oneline --graph
*   a23ecc6 (HEAD -> dev) Merged after resolving conflicts
| \
|  * cbe0f92 (main) Initialcommit in main branch
|  * 5bf21f3 initial
| /
|
| * cc4c297 Changes in dev branch
| * 86854c2 Committed in dev branch hjsx abdchj kqncuilk
| * d5d7d09 (origin/dev) Committed in dev branch
| * fc1d99d Second change
| * 08acca7 (origin/main) Initial

```

## Network graph

Timeline of the most recent commits to this repository and its network ordered by most recently pushed to.



Now our conflict is resolved we can also check on GitHub. This branch has no conflict with the base branch.



# Case of Conflict merging due to collaborator's activity-

- To make a merge conflict changing line 6 of code

Harshita-Batra-1 / Task2\_Anushka Public

Unpin Watch 0

<> Code Pull requests Actions Projects Wiki Security Insights Settings

Task2\_Anushka /  in

<> Edit file Preview changes Tabs

```
1  #include<iostream>
2  using namespace std;
3  //I'm commenting and editing
4  //Using Loop Method
5  int main() {
6      float n,q;          //This is to make a merge conflict
7      cin>>n>>q;
8      int sum=0; //declaring and initializing
9
10     int prod=1; //declaring and initializing
11     for(int i=1;i<=n;i++){
12         if (q==1){ //using if else
13             sum = sum +i;
14         }
15         else {
16             prod =prod * i;
17         }
```

# 🔗 Harshita-Batra-1 / Task2\_Anushka Public

forked from anushkam012/Task2\_Anushka

[Code](#) [Pull requests](#) **1** [Actions](#) [Projects](#) [Wiki](#) [Security](#) [Insights](#)

Task2\_Anushka /  in

[Edit file](#)

[Preview changes](#)

```
1  #include<iostream>
2  using namespace std;
3  //I'm commenting and editing
4  //Using Loop Method
5  int main() {
6      int n,q;      //This is an exapmle to make a merge conflict
7      cin>>n>>q;
8      int sum=0; //declaring and initializing
9
10     int prod=1; //declaring and initializing
11     for(int i=1;i<=n;i++){
12         if (q==1){ //using if else
13             sum = sum +i;
14         }
15         else {
16             prod =prod * i;
17         }
18     }
```



## Commit changes

Update sumOrProducts.cpp

Add an optional extended description...

- ☐ Commit directly to the `main` branch.
- ☒ Create a **new branch** for this commit and start a pull request. [Learn more about pull requests](#)

Harshita-2

**Propose changes**

Cancel

```

BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/t2 (master)
$ cd Task2_Anushka

BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/t2/Task2_Anushka (main)
$ branch
bash: branch: command not found

BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/t2/Task2_Anushka (main)
$ git branch
* main

BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/t2/Task2_Anushka (main)
$ git branch conflictdemo

BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/t2/Task2_Anushka (main)
$ branch
bash: branch: command not found

BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/t2/Task2_Anushka (main)
$ git branch
  conflictdemo
* main

BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/t2/Task2_Anushka (main)
$ git checkout conflictdemo
Switched to branch 'conflictdemo'

BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/t2/Task2_Anushka (conflictdemo)
$ ls
sumOrProducts.cpp

BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/t2/Task2_Anushka (conflictdemo)
$ vi sumOrProducts.cpp

BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/t2/Task2_Anushka (conflictdemo)
$ git status
On branch conflictdemo
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:   sumOrProducts.cpp

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

```

MINGW64; c:/Users/Harshita batra/Desktop/t2/Task2\_Anushka

```


#include<iostream>
using namespace std;
//I'm commenting and editing
int main() {
    float n,q; // changing to have merge conflict
    cin>>n>>q;
    int sum=0; //declaring and initializing

    int prod=1; //declaring and initializing
    for(int i=1;i<=n;i++){ //using for loop
        if (q==1){ //using if else
            sum = sum +i;
        }
        else {
            prod =prod * i;
        }
    }
    if (q ==1){
        cout<<sum<<endl;
    }
    else if(q==2){
        cout<<prod<<endl;
    }
}

```

sumOrProducts.cpp[+] [dos] (23:24 22/05/2022)

:wq|

 MINGW64:/c/Users/Harshita batra/Desktop/t2/Task2\_Anushka

```
BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/t2/Task2_Anushka (conflictdemo)
$ git add .
```

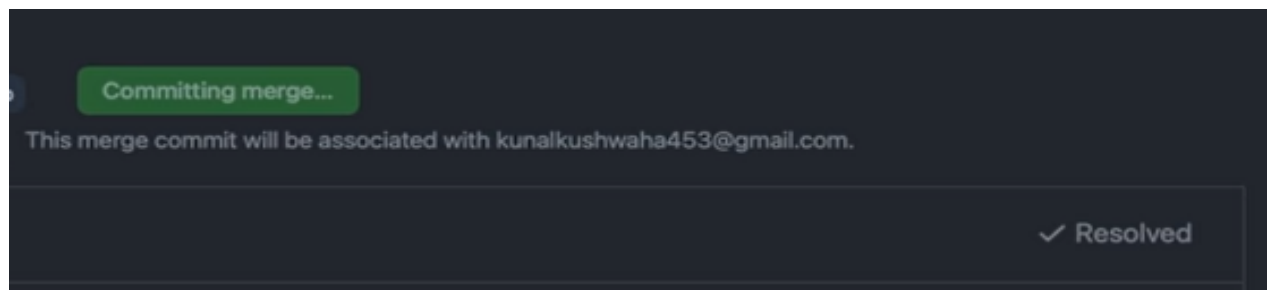
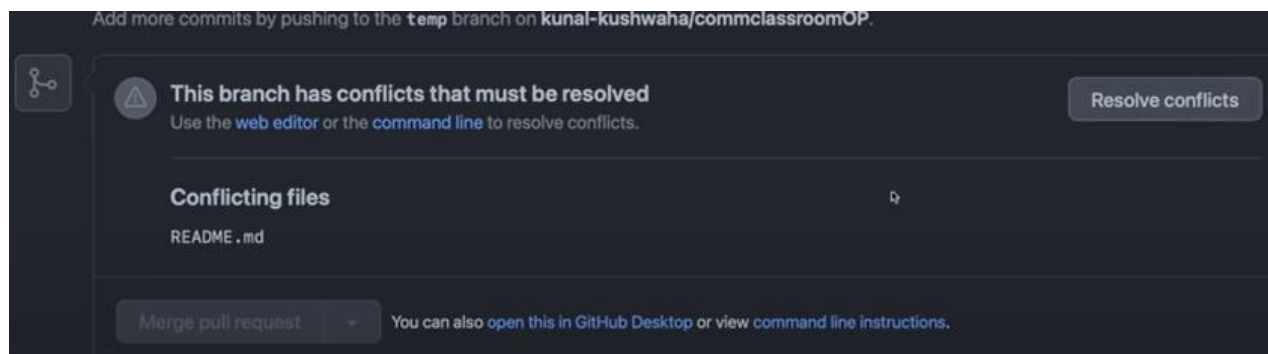
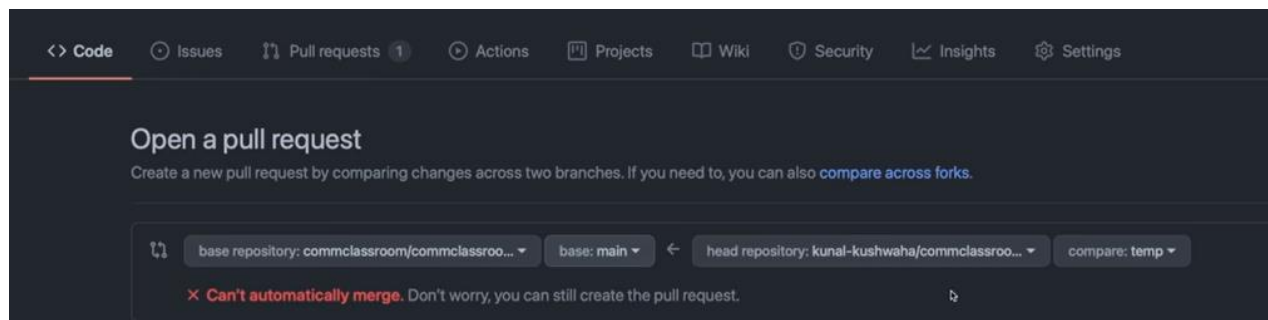
```
BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/t2/Task2_Anushka (conflictdemo)
$ git commit -m"Conflict demo"
[conflictdemo 8af8cf8] Conflict demo
1 file changed, 1 insertion(+), 1 deletion(-)
```

```
BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/t2/Task2_Anushka (conflictdemo)
$ cat sumOrProducts.cpp
#include<iostream>
using namespace std;
//I'm commenting and editing
int main() {
    float n,q;// changing to have merge conflict
    cin>>n>>q;
    int sum=0; //declaring and initializing

    int prod=1; //declaring and initializing
    for(int i=1;i<=n;i++){ //using for loop
        if (q==1){ //using if else
            sum = sum +i;
        }
        else {
            prod =prod * i;
        }
    }
    if (q ==1){
        cout<<sum<<endl;
    }
    else if(q==2){
        cout<<prod<<endl;
    }
    else{
        cout<<-1;
    }
}
```

```
BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/t2/Task2_Anushka (conflictdemo)
$ git branch
* conflictdemo
main
```

```
BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/t2/Task2_Anushka (conflictdemo)
$ git push origin conflictdemo
Enumerating objects: 12, done.
Counting objects: 100% (12/12), done.
Delta compression using up to 4 threads
Compressing objects: 100% (6/6), done.
```



Overriding of changes made by developers which was causing conflict results in finally successfully merging it.

# Harshita 2 1 #5

Merged

Harshita-Batra-1 merged 2 commits into `anushkam012:main` from `Harshita-Batra-1:Harshita-2-1` now

Conversation 0

Commits 2

Checks 0

Files changed 1



Harshita-Batra-1 commented 15 seconds ago

Collaborator

No description provided.



Harshita-Batra-1 added 2 commits 29 minutes ago



Update `sumOrProducts.cpp`

Verified



Create `sumOrProducts.cpp`

Verified



Harshita-Batra-1 merged commit `879562f` into `anushkam012:main` now

## **Aim: Reset and Revert**

### **Git Reset -**

Git reset is a powerful command that is used to undo local changes to the state of a Git repo. Git reset operates on "The Three Trees of Git". These trees are the Commit History (HEAD), the Staging Index, and the Working Directory.

The easiest way to undo the last Git commit is to execute the "git reset" command with the "--soft" option that will preserve changes done to your files.

Git reset --hard, which will completely destroy any changes and remove them from the local directory.

### **Git Revert -**

Git revert Make changes in file in git and check status to discard changes in working directory

```
BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/Revert (master|REVERTING)
$ git revert d477467366b357dbefc6d71cb053aeda5bbfbe4e
```

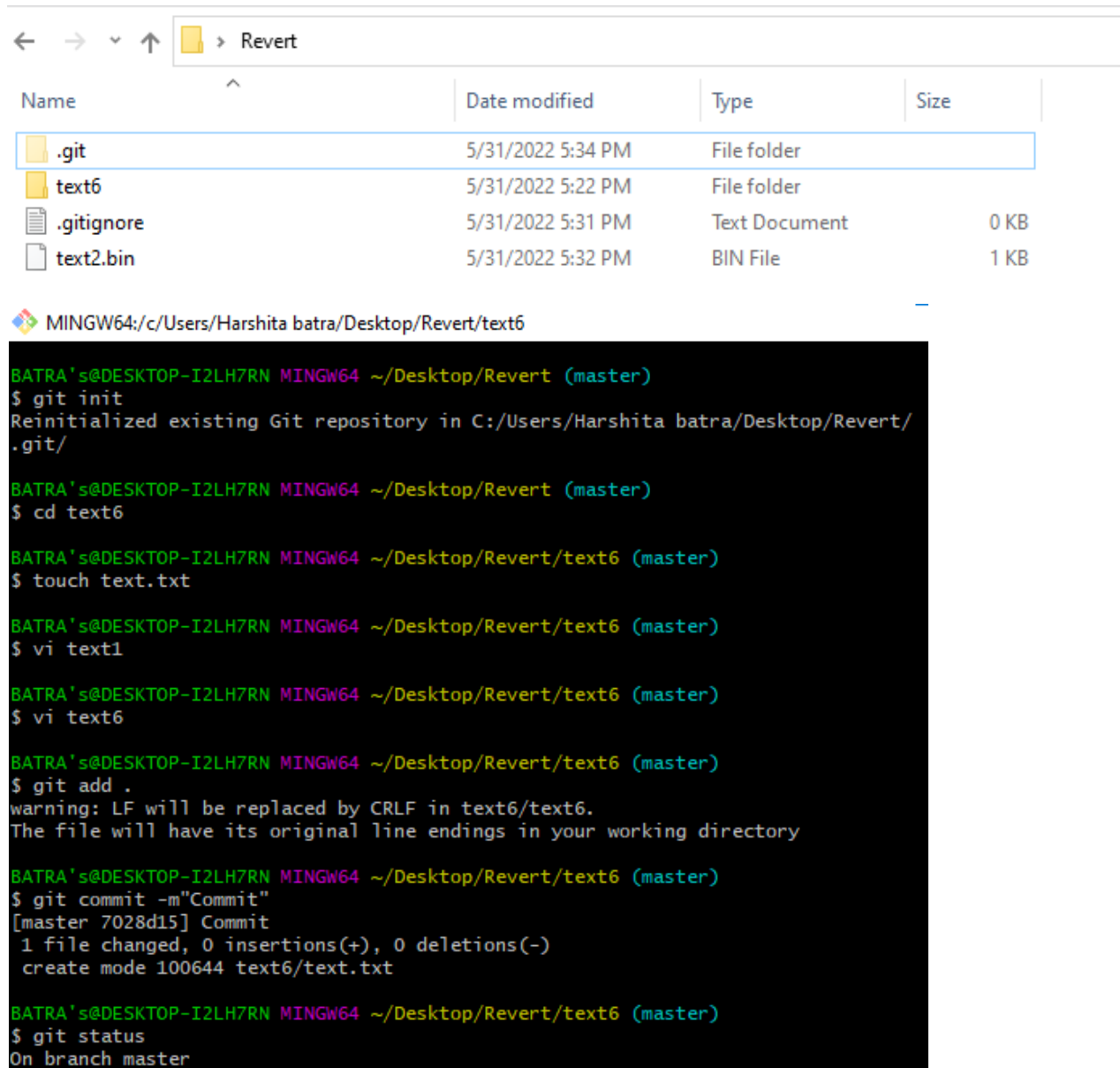
## **Theory:**

It is fairly usual for developers to build branches, add files, and stage them for commits when they are ready while working on a collaborative project. However, in certain circumstances, you may discover that the modifications you made were ineffective. You changed several files and added and removed a lot of lines, but you want to go back.

In brief, you wish to undo the modifications you just made and return to the original files.



This approach is known as "reset to HEAD," and it is a very useful tool for engineers.



Name	Date modified	Type	Size
.git	5/31/2022 5:34 PM	File folder	
text6	5/31/2022 5:22 PM	File folder	
.gitignore	5/31/2022 5:31 PM	Text Document	0 KB
text2.bin	5/31/2022 5:32 PM	BIN File	1 KB

```
MINGW64:/c:/Users/Harshita batra/Desktop/Revert/text6
BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/Revert (master)
$ git init
Reinitialized existing Git repository in C:/Users/Harshita batra/Desktop/Revert/.git/

BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/Revert (master)
$ cd text6

BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/Revert/text6 (master)
$ touch text.txt

BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/Revert/text6 (master)
$ vi text1

BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/Revert/text6 (master)
$ vi text6

BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/Revert/text6 (master)
$ git add .
warning: LF will be replaced by CRLF in text6/text6.
The file will have its original line endings in your working directory

BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/Revert/text6 (master)
$ git commit -m"Commit"
[master 7028d15] Commit
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 text6/text.txt

BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/Revert/text6 (master)
$ git status
On branch master
```

Here I modified the text6 and made new text.txt file. I added two lines in text.txt too.

```
BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/Revert/text6 (master)
$ git reset --soft HEAD~1

BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/Revert/text6 (master)
$ vi text1.txt

BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/Revert/text6 (master)
$ vi text1

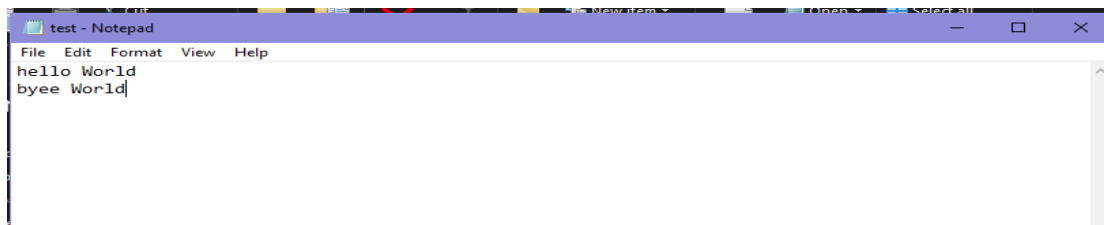
BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/Revert/text6 (master)
$ git add .

BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/Revert/text6 (master)
$ git commit -m"Seconf"
[master d477467] Seconf
1 file changed, 1 deletion(-)
delete mode 100644 text6/text6

BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/Revert/text6 (master)
$ git status
On branch master
```


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

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



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**

 MINGW64:/c/Users/Harshita batra/Desktop/Revert

```
BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/Revert (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   text6/text1
        modified:   text6/text1.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:   text6/text1

BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/Revert (master)
$ git add .
warning: LF will be replaced by CRLF in text6/text1.
The file will have its original line endings in your working directory

BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/Revert (master)
$ git commit -m"Second Commit"
[master 01d4361] Second Commit
 2 files changed, 3 insertions(+), 1 deletion(-)

BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/Revert (master)
$ git reset --hard HEAD~1
HEAD is now at 78c1e76 First commit

BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/Revert (master)
$ touch text2.bin

BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/Revert (master)
$ touch .gitignore

BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/Revert (master)
$ vi text2.bin

BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/Revert (master)
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .gitignore
        text2.bin

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

BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/Revert (master)
$ cat text2.bin
```

```
BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/Revert (master)  
$ cat text2.bin  
Hello
```

Now added lines “bye World” removed as we can see.

# PROJECT - 2



- ❖ Created a distributed repository
- ❖ Added team members there
- ❖ Added codes there so that they can fork.

MINGW64:/c/Users/Harshita batra/Desktop/Task2

```
BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/Task2 (master)
$ git init
Initialized empty Git repository in C:/Users/Harshita batra/Desktop/Task2/.git/

BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/Task2 (master)
$ git status
On branch master

No commits yet

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

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

BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/Task2 (master)
$ git add .

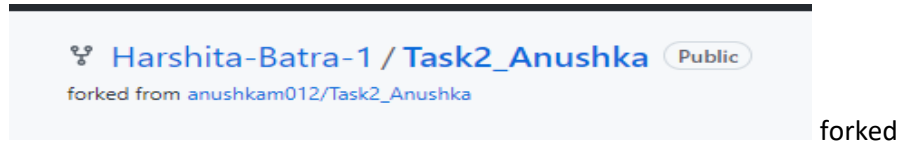
BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/Task2 (master)
$ git commit -m "Initial"
[master (root-commit) 0261e8b] Initial
 1 file changed, 19 insertions(+)
 create mode 100644 main.cpp

BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/Task2 (master)
$ git remote add origin https://github.com/Harshita-Batra-1/Task2_Harshita-Batra.git

BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/Task2 (master)
$ git branch -M main

BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/Task2 (main)
$ git push -u origin main
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 463 bytes | 463.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/Harshita-Batra-1/Task2_Harshita-Batra.git
 * [new branch]      main -> main
Branch 'main' set up to track remote branch 'main' from 'origin'.
```

## ❖ Forked team member's repository



## ❖ Made changes in code –

```
MINGW64:/c/Users/Harshita batra/Desktop/SCMtask

BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/SCMtask (master)
$ git init
Initialized empty Git repository in C:/Users/Harshita batra/Desktop/SCMtask/.git/

BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/SCMtask (master)
$ git clone https://github.com/Harshita-Batra-1/Task2_Anushka.git
Cloning into 'Task2_Anushka'...
remote: Enumerating objects: 6, done.
remote: Counting objects: 100% (6/6), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 6 (delta 1), reused 6 (delta 1), pack-reused 0
Receiving objects: 100% (6/6), done.
Resolving deltas: 100% (1/1), done.

BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/SCMtask (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    Task2_Anushka/

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

```
#include<iostream>
using namespace std;
//Hello I'm adding comments
int main() {
    int n,q;
    cin>>n>>q;
    int sum=0; //initializing

    int prod=1; //initializing
    for(int i=1;i<=n;i++){ //using for loop
        if (q==1){
            sum = sum +i;
        }
        else {
            prod =prod * i;
        }
    }
    if (q ==1){
        cout<<sum<<endl;
    }
    else if(q==2){
        cout<<prod<<endl;
    }
}
```

sumOrProducts.cpp[+] [dos] (18:52 22/05/2022)

3,27 Top

:wq

```
BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/SCMtask (master)
$ ls
Task2_Anushka/  sumOrProducts.cpp

BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/SCMtask (master)
$ cd ^C

BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/SCMtask (master)
$ cd Task2_Anushka/

BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/SCMtask/Task2_Anushka (main)
$ vi ^C

BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/SCMtask/Task2_Anushka (main)
$ vi sumOrProducts.cpp

BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/SCMtask/Task2_Anushka (main)
$ ^C

BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/SCMtask/Task2_Anushka (main)
$ vi sumOrProducts.cpp

BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/SCMtask/Task2_Anushka (main)
$ cat sumOrproducts.cpp
#include<iostream>
using namespace std;
//Hello I'm adding comments
int main() {
    int n,q;
    cin>>n>>q;
    int sum=0; //initializing

    int prod=1; //initializing
    for(int i=1;i<=n;i++){ //using for loop
        if (q==1){
            sum = sum +i;
        }
        else {
            prod =prod * i;
        }
    }
    if (q ==1){
        cout<<sum<<endl;
    }
    else if(q==2){
        cout<<prod<<endl;
    }
    else{
        cout<<-1;
    }
}
```



## ❖ Pushed it

```
BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/SCMtask/Task2_Anushka (main)
$ git push
Everything up-to-date

BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/SCMtask/Task2_Anushka (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:   sumOrProducts.cpp

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

BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/SCMtask/Task2_Anushka (main)
$ git add .

BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/SCMtask/Task2_Anushka (main)
$ git commit -m"Added comments"
[main f4691de] Added comments
 1 file changed, 4 insertions(+), 4 deletions(-)

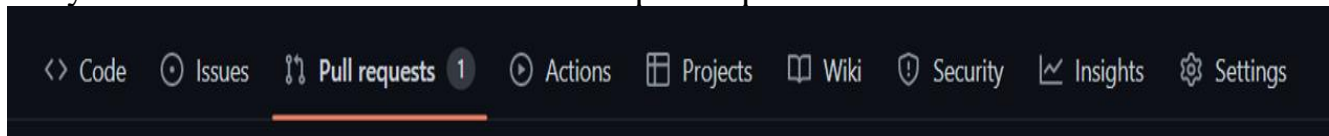
BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/SCMtask/Task2_Anushka (main)
$ git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 384 bytes | 128.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/Harshita-Batra-1/Task2_Anushka.git
   3686126..f4691de  main -> main
```

Showing 1 changed file with 4 additions and 4 deletions.

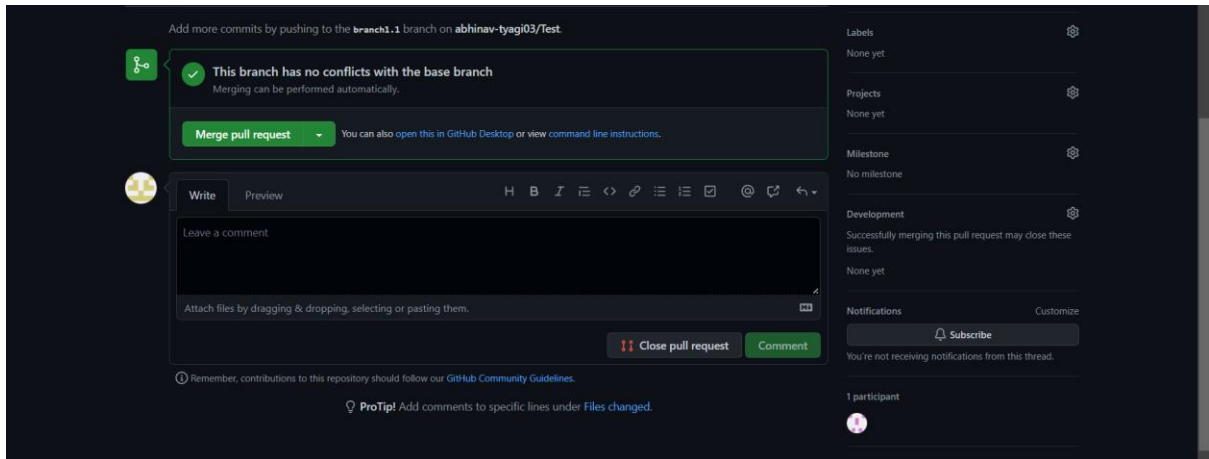
```
8 sumOrProducts.cpp
... @@ -1,13 +1,13 @@
1 1 #include<iostream>
2 2 using namespace std;
3 -
3 + //Hello I'm adding comments
4 4 int main() {
5 5     int n,q;
6 6     cin>>n>>q;
7 - int sum=0;
7 + int sum=0; //initializing
8 8
9 - int prod=1;
10 - for(int i=1;i<=n;i++){
9 + int prod=1; //initializing
10 + for(int i=1;i<=n;i++){ //using for loop
11 11     if (q==1){
12 12         sum = sum +i;
13 13     }
...
↓
```

To create a pull request on a team member's repository and close requests by any other team members as a maintainer follow the procedure given below:-

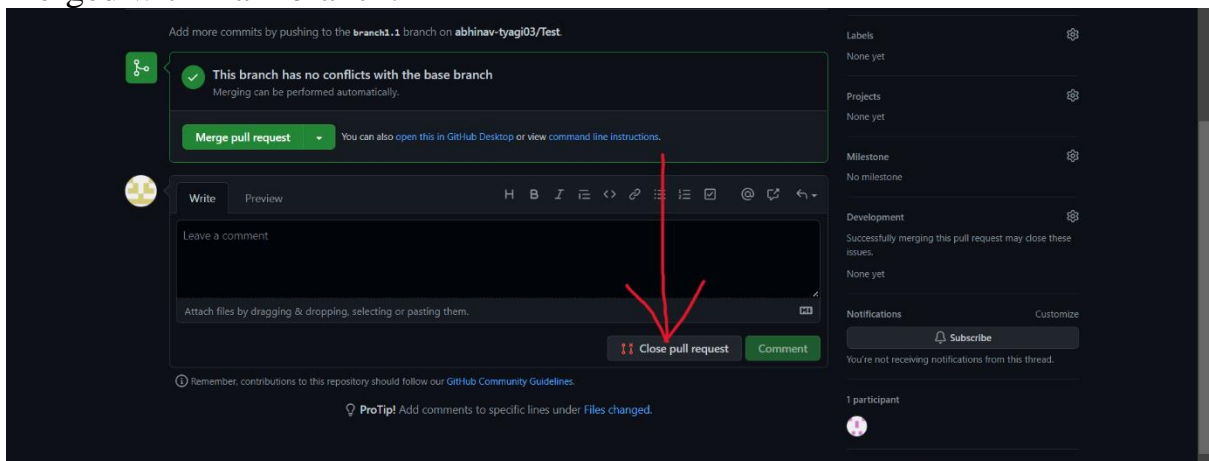
- Do the required changes in the repository, add and commit these changes in the local repository in a new branch.
- Push the modified branch using `git push origin branchname`.
- Open a pull request by following the procedure from the above experiment.
- The pull request will be created and will be visible to all the team members.
- Ask your team member to login to his/her Github account.
- They will notice a new notification in the pull request menu.



- Click on it. The pull request generated by you will be visible to them.
- Click on the pull request. Two option will be available, either to close the pull request or Merge the request with the main branch.
- By selecting the merge branch option the main branch will get updated for all the team members.



- By selecting close the pull request the pull request is not accepted and not merged with main branch.



- The process is similar to closing and merging the pull request by you. It simply includes an external party to execute.
- The result of merging the pull request is shown below.
- Thus, we conclude opening and closing of pull request. We also conclude merging of the pull request to the main branch.



## Pull request sent to my team member

Harshita-Batra-1 / Task2\_Anushka Public

forked from anushkam012/Task2\_Anushka

[Code](#) [Pull requests](#) [Actions](#) [Projects](#) [Wiki](#) [Security](#) [Insights](#) [Settings](#)

main 2 branches 0 tags

[Go to file](#) [Add file](#) [Code](#)

This branch is 1 commit ahead of anushkam012:main. [Contribute](#) [Fetch upstream](#)


Harshita-Batra-1 Added comments

sumOrProducts.cpp Added 36 seconds ago


Help people interested in this repository understand your project

[Open pull request](#) [Add a README](#)

This branch is 1 commit ahead of anushkam012:main.  
Open a pull request to contribute your changes upstream.

 base repository: anushkam012/Task2\_Anushka base: main ← head repository: Harshita-Batra-1/Task2\_Anushka compare: main

✓ **Able to merge.** These branches can be automatically merged.





Added comments

WritePreview


HBI≡<>🔗☰☷☑@🔗↩




Added comments

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

☒ Allow edits by maintainers 

Create pull request

 Remember, contributions to this repository should follow our [GitHub Community Guidelines](#).

<input type="checkbox"/>	 0 Open ✓ 2 Closed
<input type="checkbox"/>	 <b>Update main.cpp</b> #2 by AreenSiwach was merged 9 days ago
<input type="checkbox"/>	 <b>updating main.cpp</b> #1 by anushkam012 was merged 9 days ago

## ❖ Forked another team member's repository

MINGW64:/c/Users/Harshita batra/Desktop/task2/Task2\_Areen

```
BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/task2 (master)
$ git init
Initialized empty Git repository in C:/Users/Harshita batra/Desktop/task2/.git/

BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/task2 (master)
$ git clone https://github.com/Harshita-Batra-1/Task2_Areen.git
Cloning into 'Task2_Areen'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.

BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/task2 (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    Task2_Areen/

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

BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/task2 (master)
$ ls
Task2_Areen/

BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/task2 (master)
$ cd ^C

BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/task2 (master)
$ cd Task2_Areen/

BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/task2/Task2_Areen (main)
$ ls
'Mirror Number Pattern.cpp'

BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/task2/Task2_Areen (main)
$ ^C

BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/task2/Task2_Areen (main)
$ vi Mirror Number Pattern.cpp
3 files to edit

BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/task2/Task2_Areen (main)
$ cat Mirror Number Pattern.cpp
Hello I'm editing
```

```

BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/task2/Task2_Areen (main)
$ git push
Everything up-to-date

BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/task2/Task2_Areen (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

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

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

BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/task2/Task2_Areen (main)
$ git add .
warning: LF will be replaced by CRLF in Mirror.
The file will have its original line endings in your working directory

BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/task2/Task2_Areen (main)
$ git commit -m"intial"
[main ca88032] intial
1 file changed, 1 insertion(+)
create mode 100644 Mirror

BATRA's@DESKTOP-I2LH7RN MINGW64 ~/Desktop/task2/Task2_Areen (main)
$ git push
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 310 bytes | 310.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/Harshita-Batra-1/Task2_Areen.git
a25cddb..ca88032  main -> main

```

## ❖ Sent the pull request

**Able to merge.** These branches can be automatically merged.

Discuss and review the changes in this comparison with others. [Learn about pull requests](#)
Create pull request

1 commit
1 file changed
1 contributor

Commits on May 22, 2022
 

**intial**
 Harshita-Batra-1 committed 3 minutes ago
 ca88032

Showing 1 changed file with 1 addition and 0 deletions.
 Split Unified

1
 
 Mirror

```

...    ...    @@ -0,0 +1 @@
1      + Hello I'm editing
        
```

## ❖ Merging of pull request

🏠 Harshita-Batra-1 / Task2\_Harshita-Batra Public

<> Code Issues Pull requests Actions Projects Wiki ⓘ

🔗 main 1 branch 0 tags

👤 Harshita-Batra-1 Merge pull request #2 from AreenSiwach/main ...

📄 main.cpp Update main.cpp

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

🔗 Pull requests Actions Projects Wiki Security Insights Settings

🔗 main

📌 Commits on May 22, 2022

**Update Mirror Number Pattern.cpp**  
👤 AreenSiwach committed 24 minutes ago

**Merge pull request #1 from Harshita-Batra-1/main** ...  
👤 AreenSiwach committed 28 minutes ago

**intial**  
👤 Harshita-Batra-1 committed 1 hour ago

**Mirror Pattern**  
👤 AreenSiwach committed 2 hours ago



## Update main.cpp #2

Merged

Harshita-Batra-1 merged 1 commit into [Harshita-Batra-1:main](#) from [AreenSiwach:main](#) 35 seconds ago

Conversation 0

Commits 1

Checks 0

Files changed 1



AreenSiwach commented 7 minutes ago

Collaborator

*No description provided.*

Update main.cpp

Verified

b2f9ec2

Harshita-Batra-1 merged commit 707aa24 into [Harshita-Batra-1:main](#) now[Revert](#)☐ 0 Open ✓ 2 Closed☐ Update main.cpp

#2 by AreenSiwach was merged 9 days ago

☐ updating main.cpp

#1 by anushkam012 was merged 9 days ago

## **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 contain commits unique to the network.

A repository's graphs give you information on traffic, projects that depend on the repository, contributors and commits to the repository, and a repository's forks and network. If you maintain a repository, you can use this data to get a better understanding of who's using your repository and why they're using it.

Some repository graphs are available only in public repositories with GitHub Free:

- Pulse
- Contributors
- Traffic
- Commits
- Code frequency
- Network

## **Steps to access network graphs of respective repository**

1. On GitHub.com, navigate to the main page of the repository.
2. Under your repository name, click **Insights**.



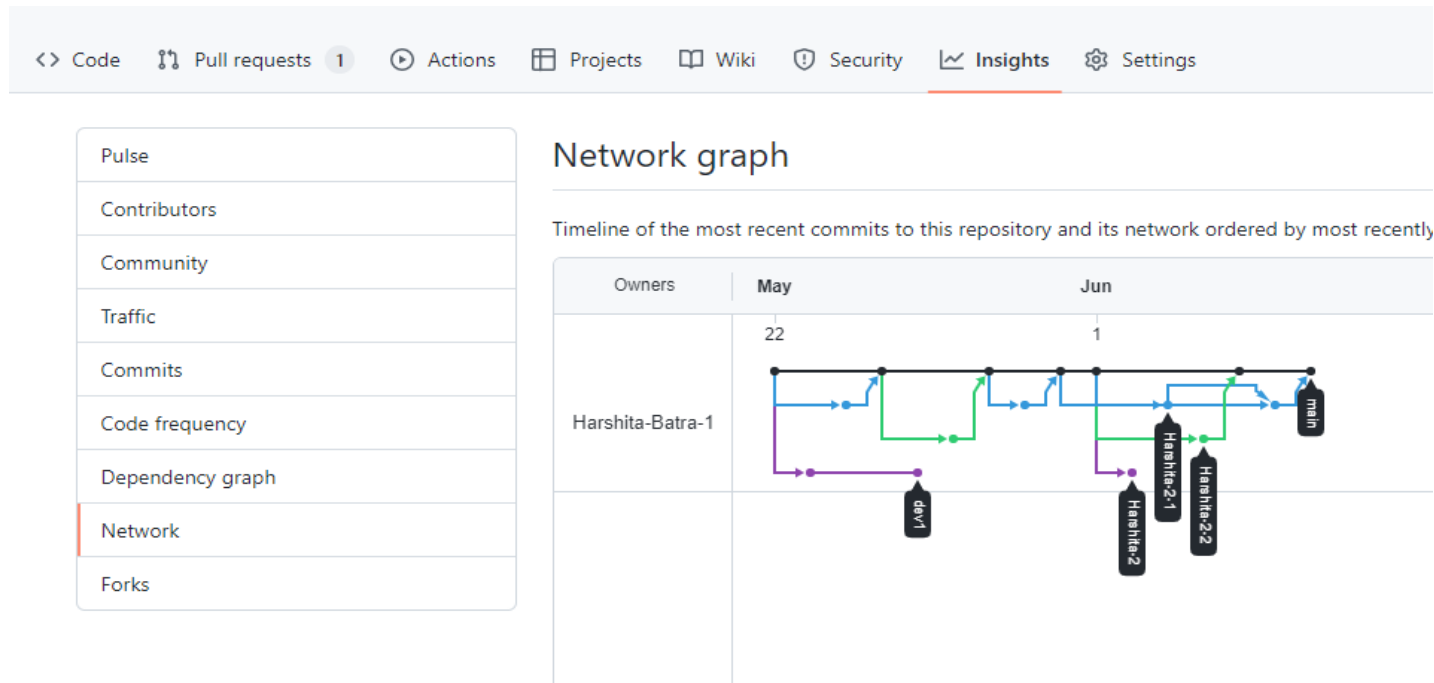
3. At the left sidebar, click on **Network**.

## Listing the forks of a repository

Forks are listed alphabetically by the username of the person who forked the repository

Clicking the number of forks shows you the full network. From there you can click "members" to see who forked the repo

1. On GitHub.com, navigate to the main page of the repository.
2. Under your repository name, click **Insights**.



Harshita-Batra-1 / Task2\_Harshita-Batra

Public

Unpin

Unwatch 1

Fork 2

Star 0

<> Code

Issues

Pull requests

Actions

Projects

Wiki

Security

Insights

Settings

Pulse

Contributors

Community

Community Standards

Traffic

Commits

Code frequency

Dependency graph

Network

Forks

April 30, 2022 – May 31, 2022

Period: 1 month

Overview

2 Active pull requests

0 Active issues

2 Merged pull requests

0 Open pull requests

0 Closed issues

0 New issues

Excluding merges, 3 authors have pushed 4 commits to main and 4 commits to all branches. On main, 0 files have changed and there have been 0 additions and 0 deletions.

2

1

1

Activate Windows

Go to Settings to activate Windows.

2 Pull requests merged by 2 people

Update main.cpp

#2 merged 9 days ago

updating main.cpp

#1 merged 9 days ago

## Viewing the dependencies of a repository

You can use the dependency graph to explore the code your repository depends on.

Almost all software relies on code developed and maintained by other developers, often known as a supply chain. For example, utilities, libraries, and frameworks. These dependencies are an integral part of your code and any bugs or vulnerabilities in them may affect your code. It's important to review and maintain these dependencies.

