**Source Code Management**

**Task 1.1**

**&**

**Task 1.2**

**&**

**Task 2**

(CS181)

Submitted by

Name-Aanchal Sharma Roll No.- 2110990009



**Department of Computer Science & Engineering**

Chitkara University Institute of Engineering and Technology, Punjab

Jan- June   
(2021-22)

|  |  |  |  |
| --- | --- | --- | --- |
| Institute/School Name | **Chitkara University Institute of Engineering and Technology** | | |
| Department Name | **Department of Computer Science & Engineering** | | |
| Programme Name | **Bachelor of Engineering (B.E.), Computer Science & Engineering** | | |
| Course Name | **Source Code Management** | Session | **2021-22** |
| Course Code | **CS181** | Semester/Batch | **2nd/2021** |
| Vertical Name | **Beta** | Group No | G01 |
| Course Coordinator | **Dr. Neeraj Singla** | | |
| Faculty Name | **Mr. Monit Kapoor** | | |

TASK 1.1

|  |  |
| --- | --- |
| S.no. | Title |
| 1. | Setting up of Git Client |
| 2. | Setting up of GitHub Account |
| 3. | Generate logs |
| 4. | Create and visualize branches |
| 5. | Git lifecycle description |

**Experiment No. 01**

**Aim:** Setting up of Git Client

**Theory:**

GIT –> It is basically used for pushing and pulling of code. We can use git and git-hub parallelly to work with multiple members or individually. We can make , edit , recreate ,copy or download any code on git hub using git.

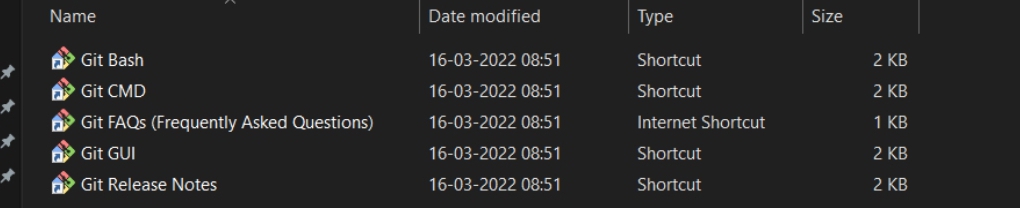
What is GIT ? –> It’s a Version Control System(VCS) -> It is a software or we can say a server by which we are able to track all the previous changes in the code.

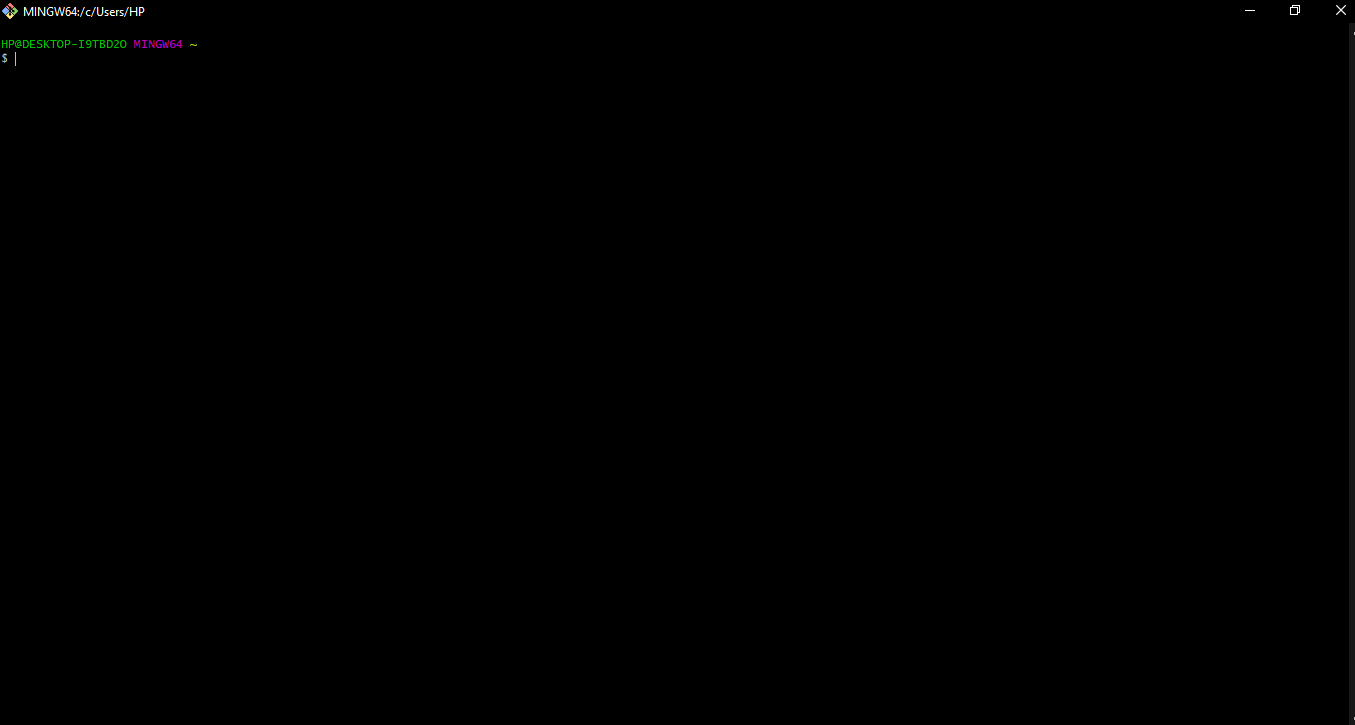
Advantages of GIT –>

**Procedure:** We can install Git on Windows, using the most official build which is available for download on the GIT’s official website or by just typing ( s c m git ) on any search engine . We can go on <https://git-scm.com/download/win> and can select the platform and bit-version to download. And after clicking on your desired bit-version or ios it will start downloading automatically.

**Snapshots of download:**







**Experiment No. 02**

**Aim:** Setting up GitHub Account

**Theory:**

**What is GitHub ->** GitHub is a website and cloud-based service (client) that helps an individual or a developers to store and manage their code. We can also track as well as control changes to our or public code.

**Advantages of GitHub ->** GitHub’s has a user-friendly interface and is easy to use .We can connect the git-hub and git but using some commands shown below in figure 001. Without GitHub we cannot use Git because it generally requires a host and if we are working for a project we need to share it will our team members, which can only be done by making a repository . Additionally , anyone can sign up and host a public code repository for free, which makes GitHub especially popular with open-source projects.

**Procedure:-**

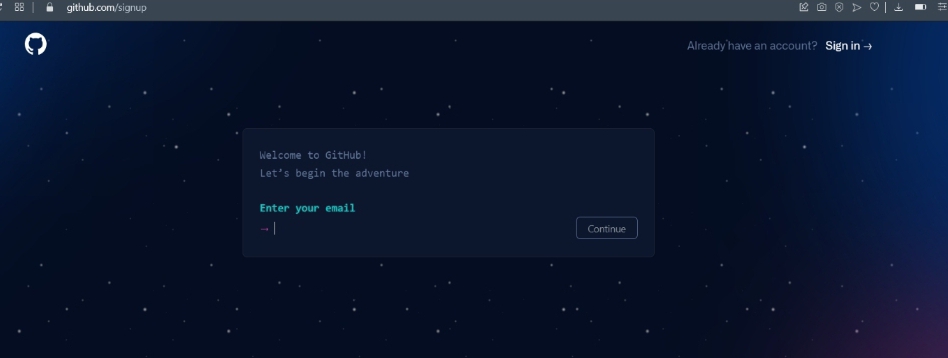
**Step1 :-**

Google (any search engine)

Search for git-hub or (<https://github.com/signup>).

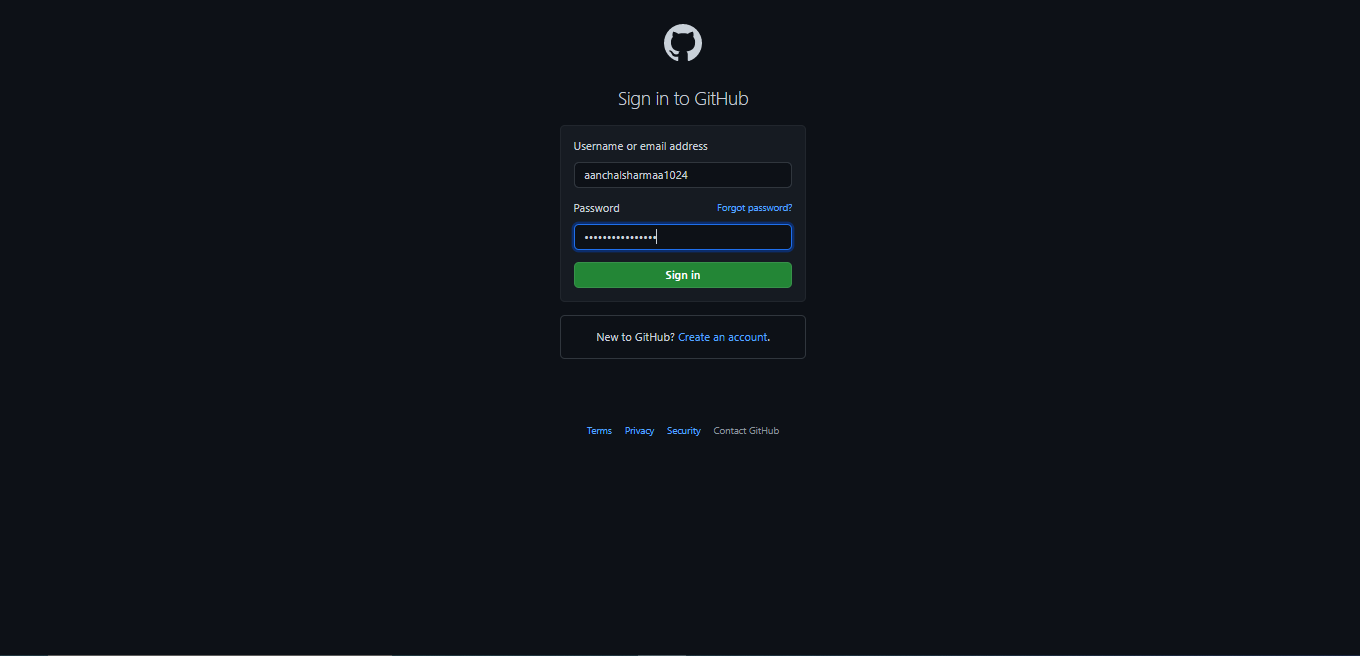
**Step2 :-**

**Snapshots** –

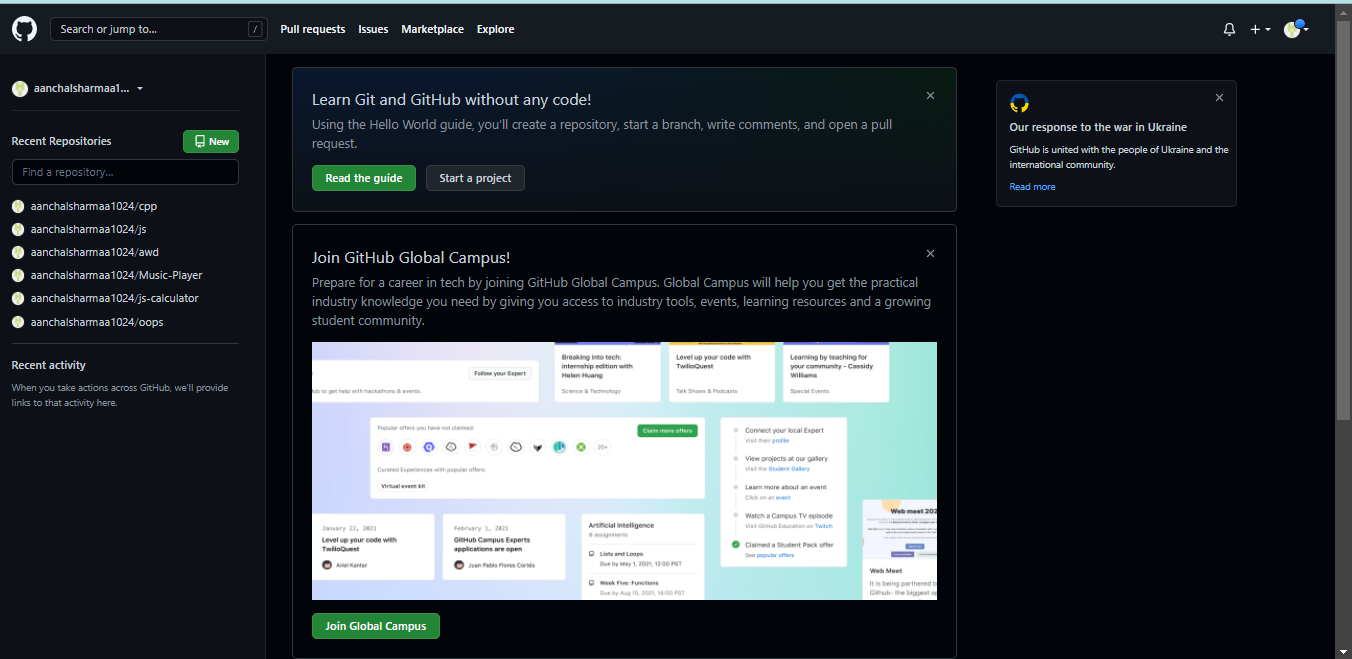


After visiting the link this type of interface will appear, if you already have account you can sign in and if not you can create.

**Sign in into GIT-HUB :-**



**Interface of GitHub :-**



**To link GitHub account with Git bash –**

**For username:-**

git config --global user.name “username in git-hub”

**For user email:-**

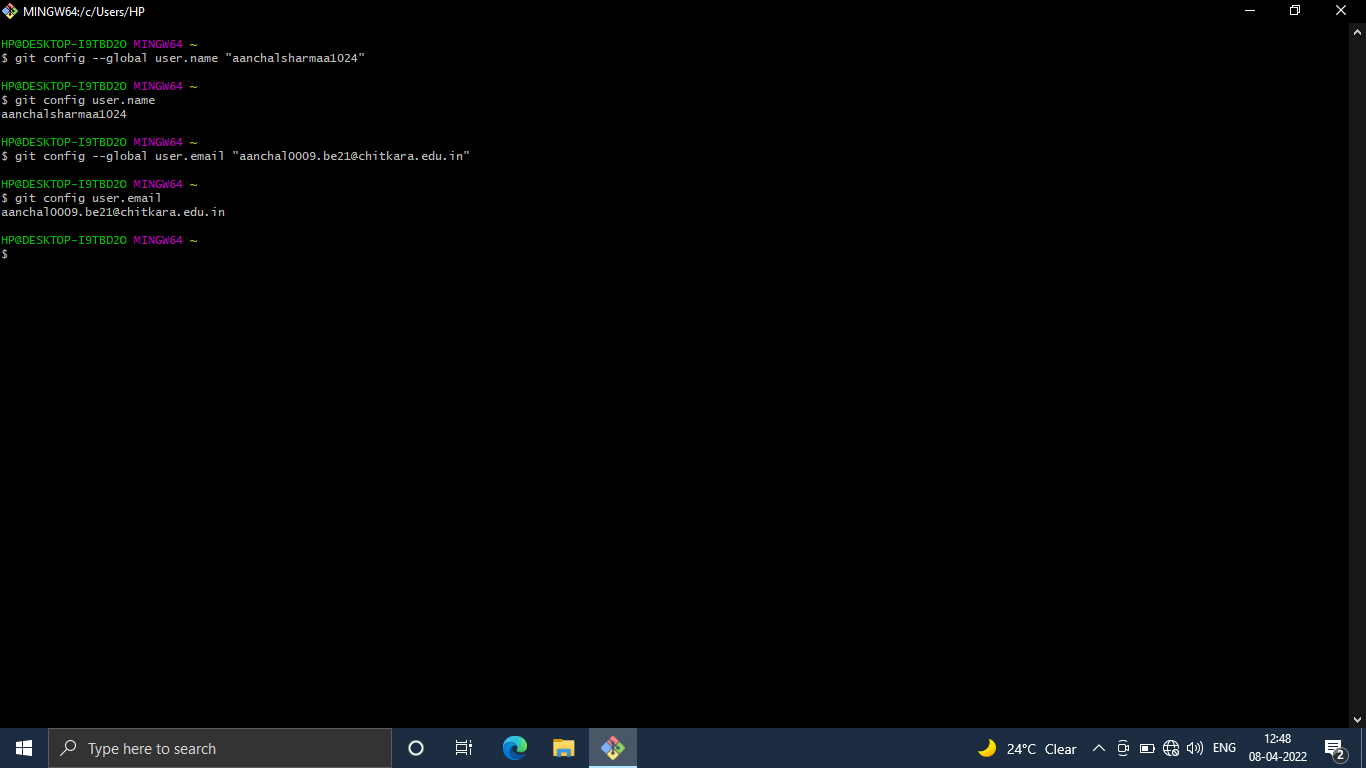
git config --global user.email “your email in git-hub”

**To verify:-**

git config user.name

git config user.email

**Snapshot :-**



**Experiment No. 03**

**Aim:** Program to Generate log

**Theory:-**

**Logs ->** Logs are nothing but the history which we can see in git by using the code git log.

It contains all the past commits, insertions and deletions in it which we can see any time.

**Why logs ->**  Logs helps to check that what were the changes in the code or any other file and by whom. It also contains the number of insertions and deletions including at which time it was changed.

**Snapshots –**



**Experiment No. 04**

**Aim:** Create and visualize branches

**Create branches :-**

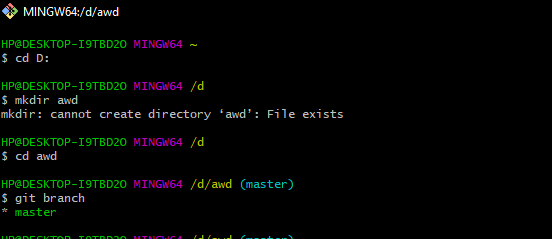
The main branch in git is called as master branch. But we can make branches out of this main master branch. All the files present in master can be shown in branch but the file which are created in branch are not shown in master branch. We can also merge both the parent (master) and child (other branches).

**Syntax:-**

1. For creating a new branch.

git branch name of branch , by default it is master branch

**Snapshots –**

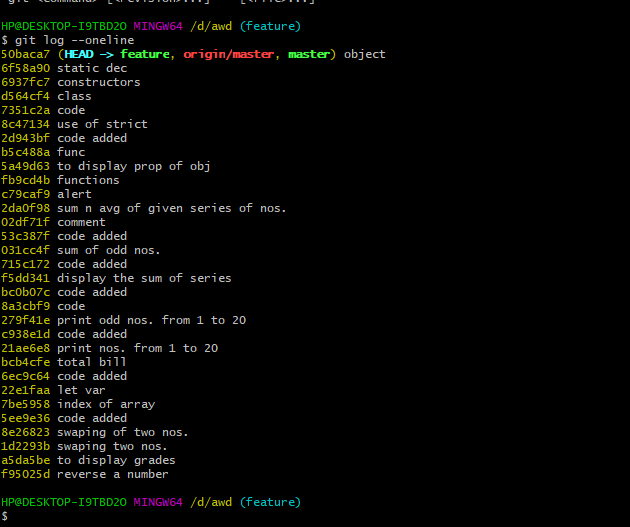


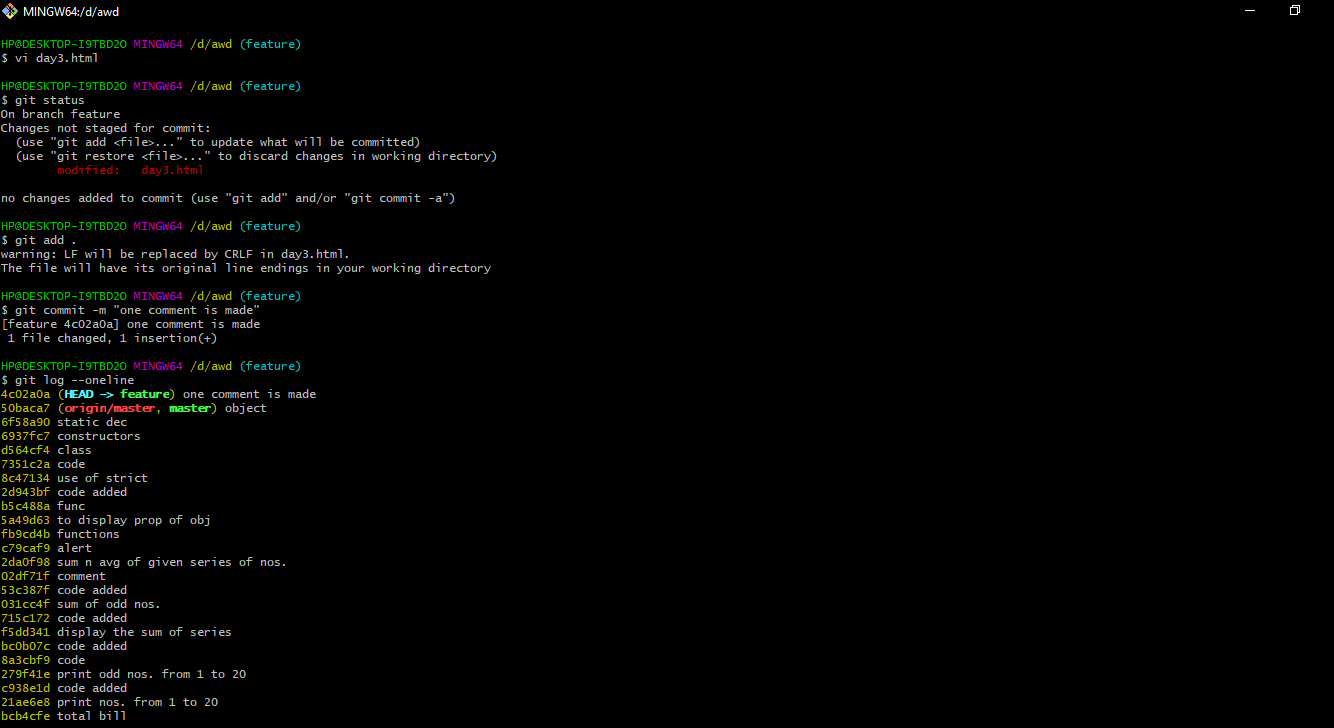
1. To change the present working branch.

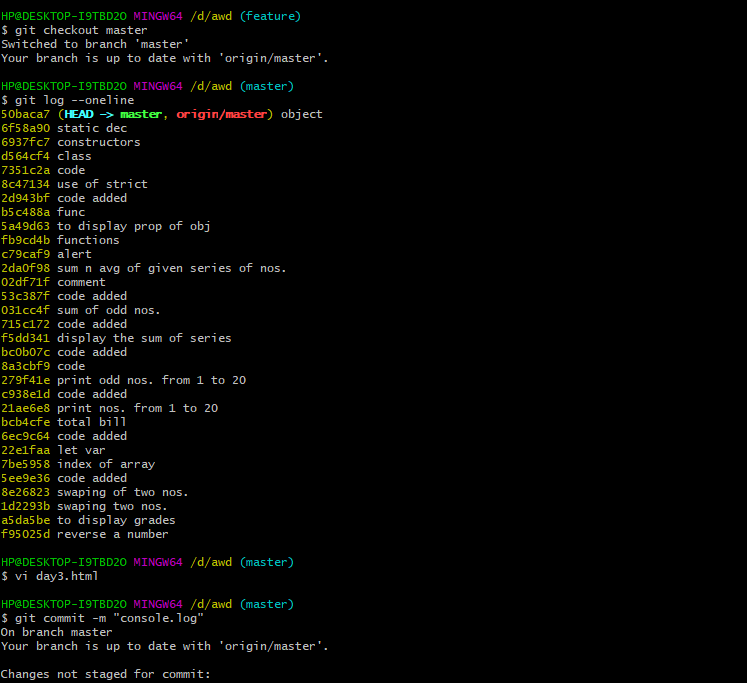
git checkout name of branch.

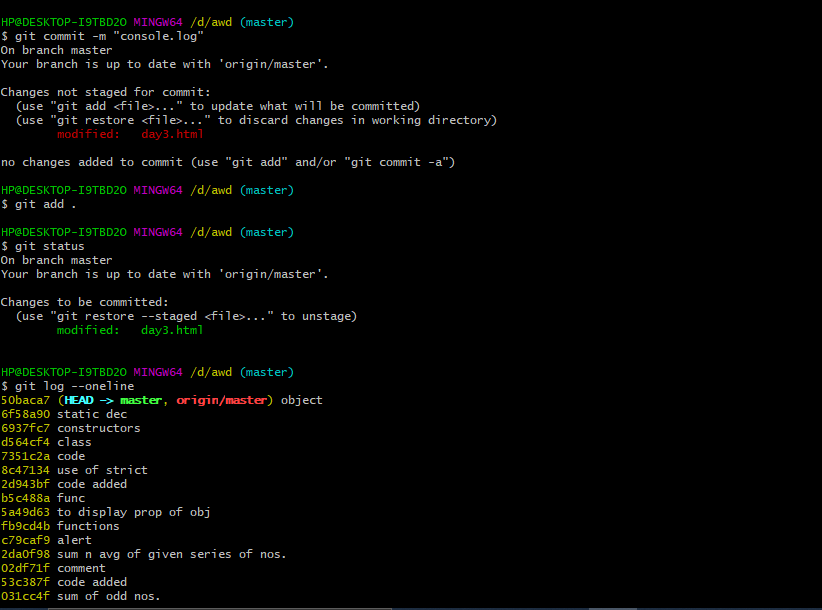


**Visualizing branches :-**









**Experiment No. 05**

**Aim:** Git lifecycle description

**Theory:**

**Stages in GIT Life Cycle ->** Files in a Git project have various stages like Creation, Modification, Refactoring, and Deletion and so on. Irrespective of whether this project is tracked by Git or not, these phases are still prevalent. However, when a project is under Git version control system, they are present in three major Git states in addition to these basic ones. Here are the three Git states:

* Working directory
* Staging area
* Git directory

**Working Directory ->**

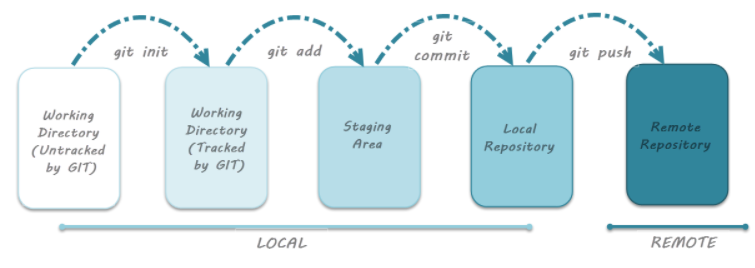
Consider a project residing in your local system. This project may or may not be tracked by Git. In either case, this project directory is called your Working directory.

### **Staging Area ->**

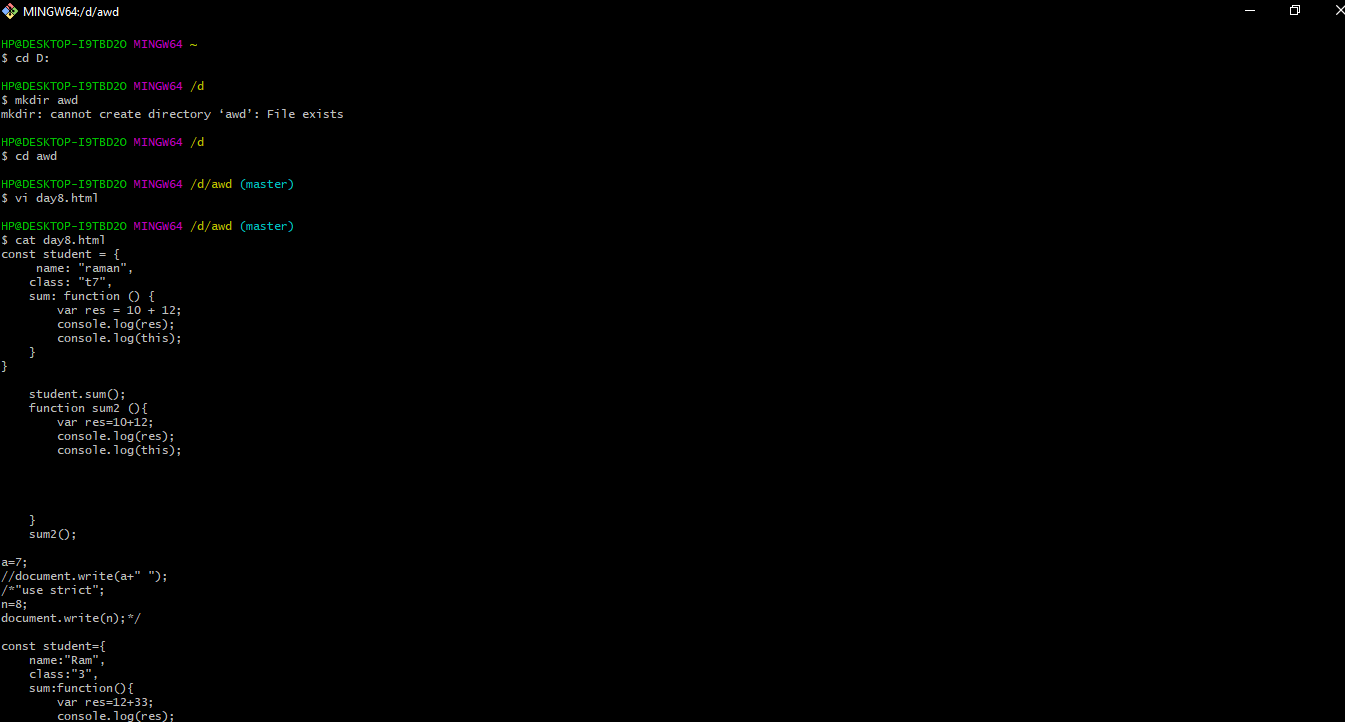
**Staging area is the playground where you group, add and organize the files to be committed to Git for tracking their versions.**

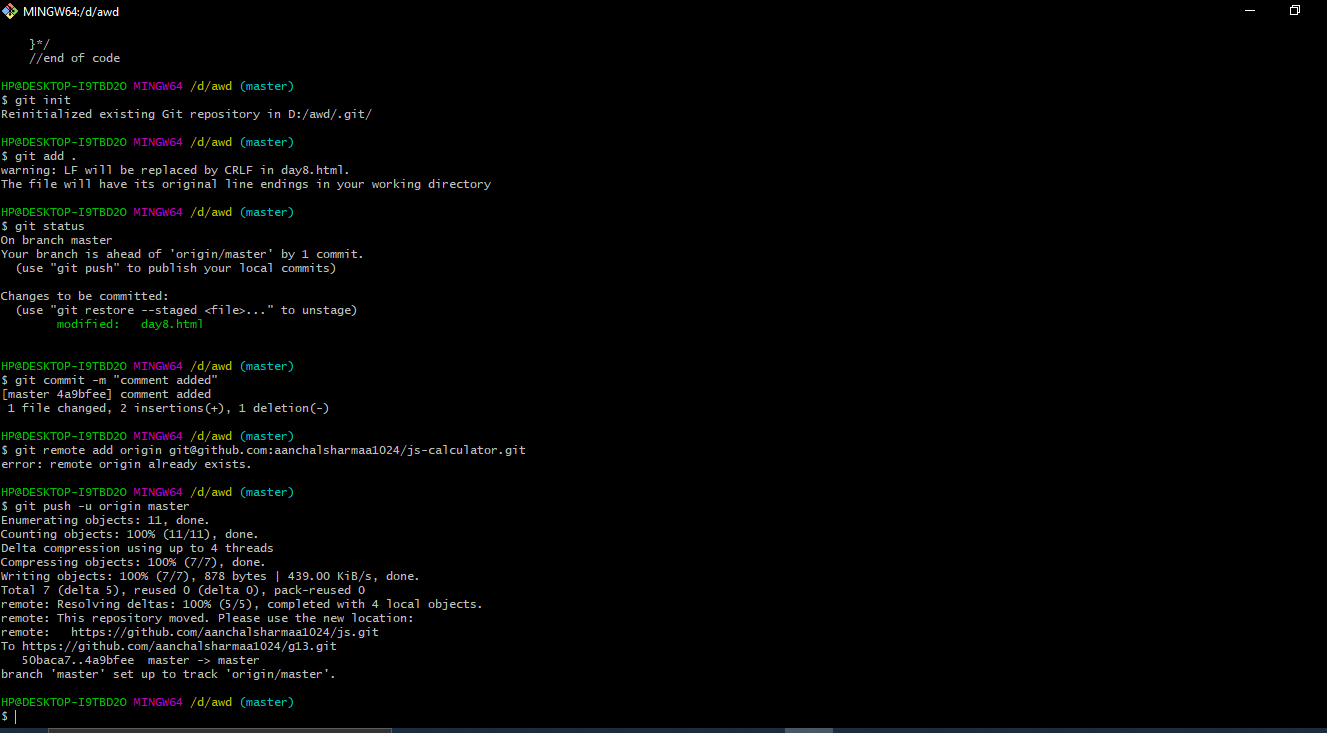
**Git Directory ->**

Now that the files to be committed are grouped and ready in the staging area, we can commit these files. So, we commit this group of files along with a commit message explaining what is the commit about. Apart from commit message, this step also records the author and time of the commit. Now, a snapshot of the files in the commit is recorded by Git. The information related to this commit is stored in the Git directory.

**Remote Repository-> means mirror or clone of the local Git repository in GitHub**. And **pushing means uploading the commits from local Git repository to remote repository hosted in GitHub.**

**Snapshots –**





TASK 1.2

|  |  |
| --- | --- |
| **S. No.** | **Title** |
| 1 | Add collaborators on GitHub Repo |
| 2 | Fork and Commit |
| 3 | Merge and Resolve conflicts created due to own activity and collaborators activity. |
| 4 | Reset and Revert |

*ADD COLLABORATORS ON GITHUB REPO*

In GitHub, we can invite other GitHub users to become collaborators to our private repositories (which expires after 7 days if not accepted, restoring any unclaimed licenses). Being a collaborator, of a personal repository you can pull (read) the contents of the repository and push (write) changes to the repository. You can add unlimited collaborators on public and private repositories.

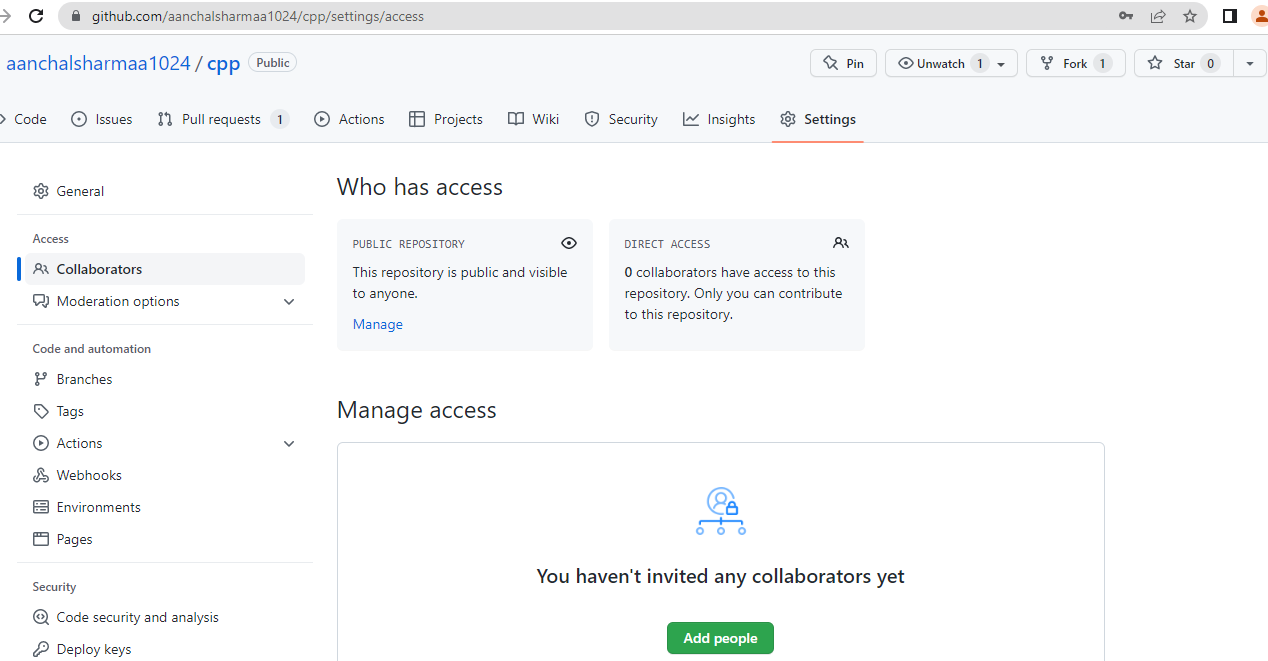
Collaborators can perform a number of actions into someone else’s personal repositories, they have gained access to. Some of them are,

1. Create, merge, and close pull requests in the repository
2. Publish, view, install the packages
3. Fork the repositories
4. Make the changes on the repositories as suggested by the Pull requests.
5. Mark issues or pull requests as duplicate
6. Create, edit, and delete any comments on commits, pull requests, and issues in the repository
7. Removing themselves as collaborators on the repositories.
8. Manage releases in the repositories.

STEPS TO ADD COLLABORATORS:

1. Navigate to the repository on Github you wish to share with your collaborator.
2. Click on the "Settings" tab on the right side of the menu at the top of the screen.

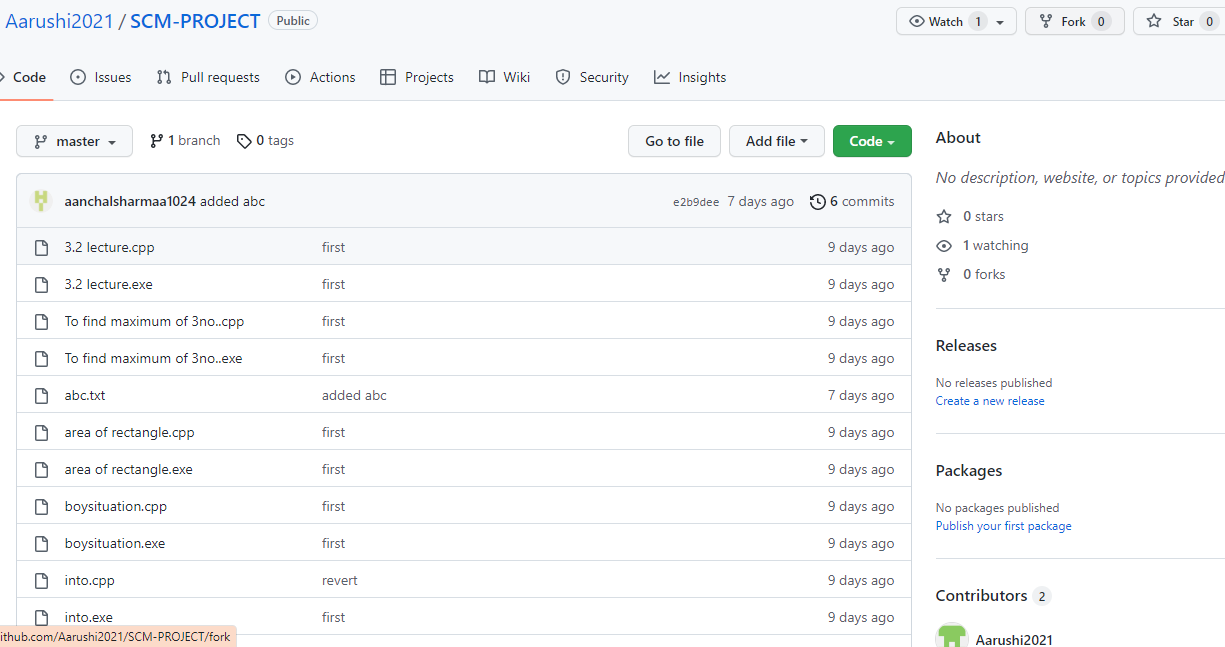
3.On the new page, click the "Collaborators" menu item on the left side of the page.



*FORK AND COMMIT*

A fork is a copy of a repository. Forking a repository allows you to freely experiment with changes without affecting the original project. Most commonly, forks are used to either propose changes to someone else's project to which you do not have write access, or to use someone else's project as a starting point for your own idea.

**STEPS TO FORK A REPO-**

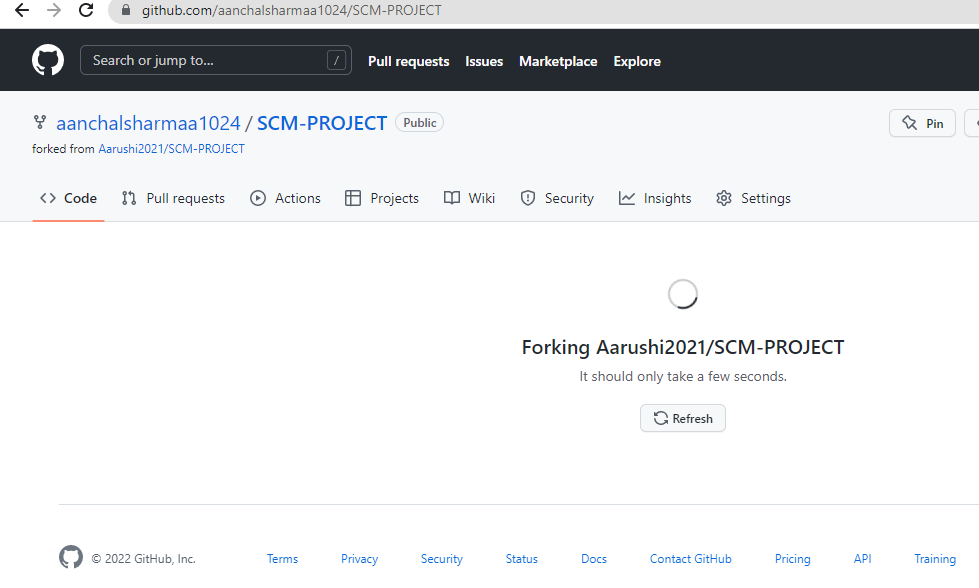


1. Go to the repository that you wish to fork.

2.Click on the option ‘Fork’ in the top right corner.

3.You now have a forked repository.

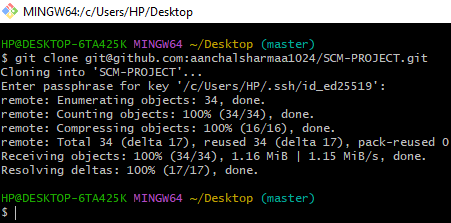




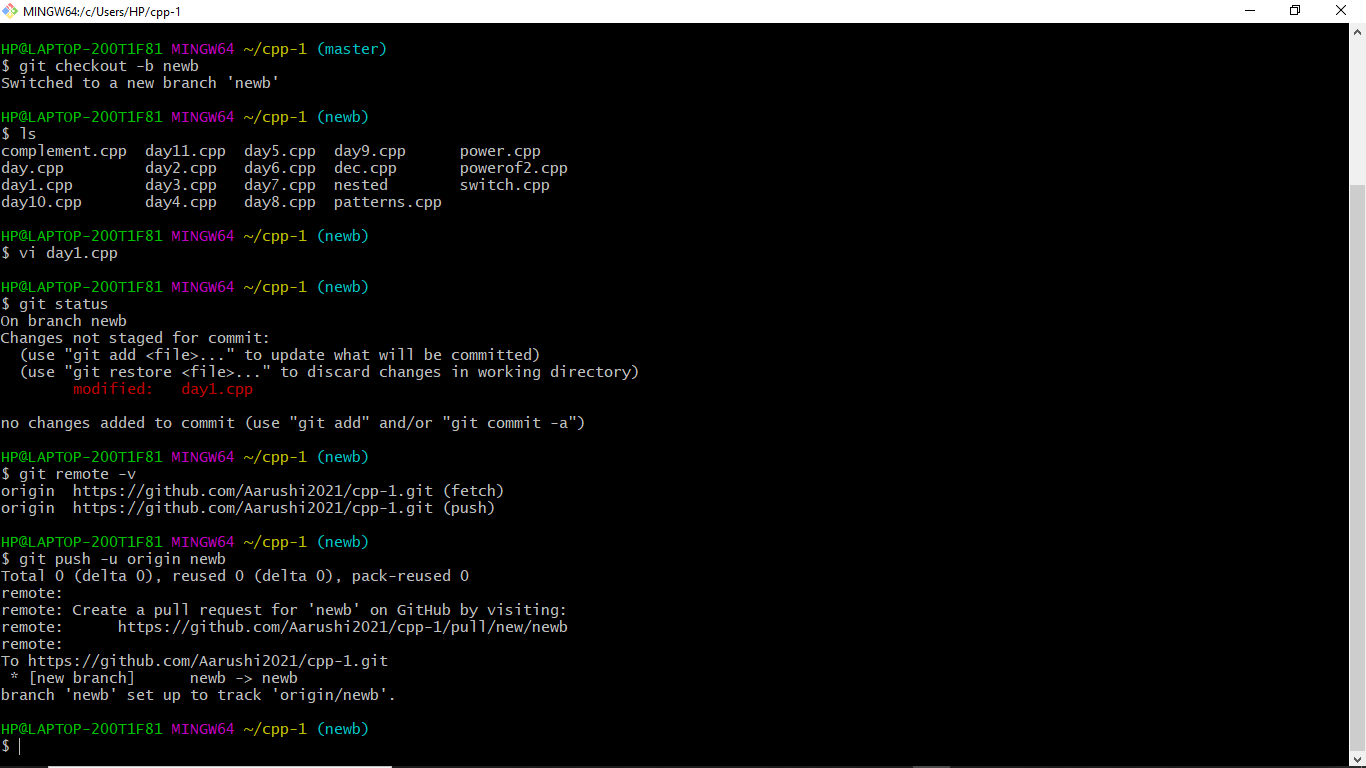
**CLONING THE REPO INTO YOUR DEVICE**

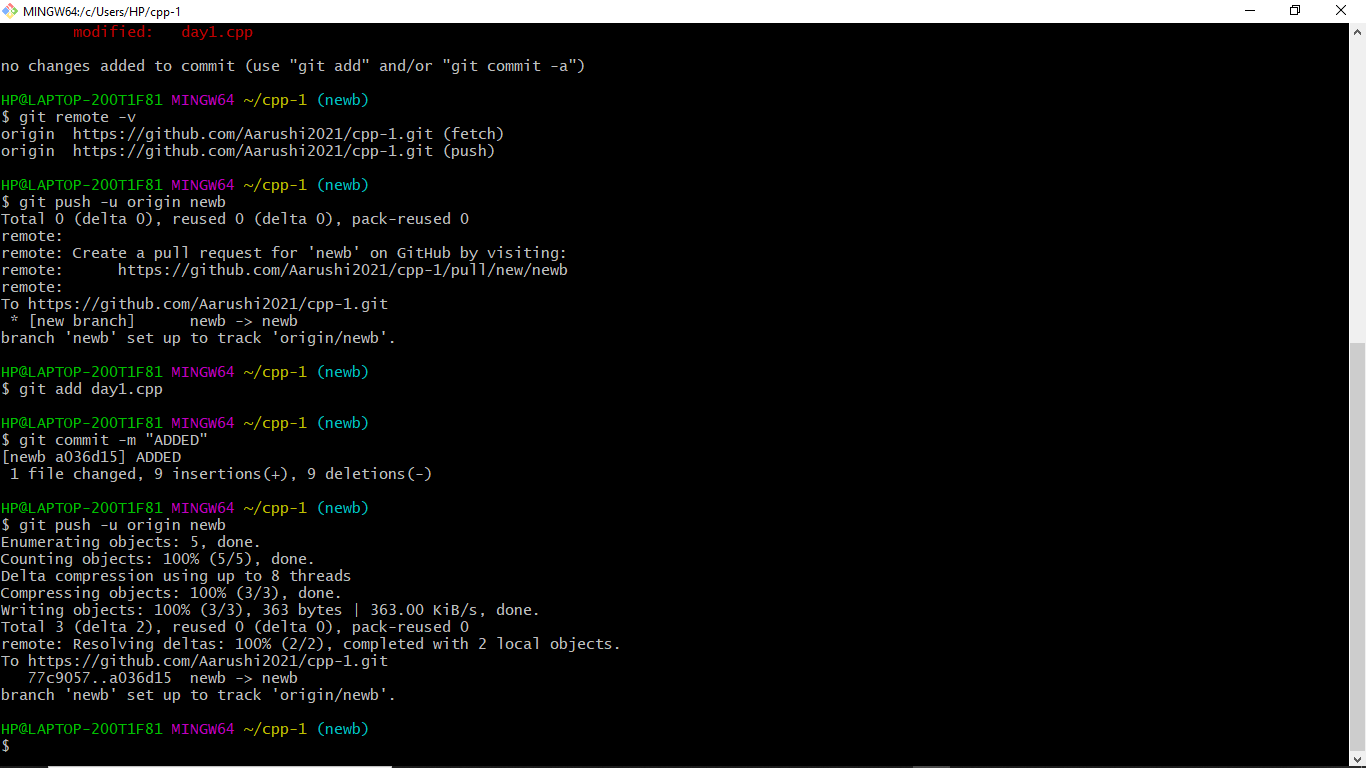
When you create a repository on GitHub.com, it exists as a remote repository. You can clone your repository to create a local copy on your computer and sync between the two locations.

1. Once you have forked the repository, you can clone it into your computer using directly the option given on github or through running git clone command in git bash.
2. Copy the URL of the forked repository
3. Open git bash and type the command “ git clone <url of the forked repository>”



**COMMITING CHANGES TO THE FORKED REPOSITORY**

1. Once you have cloned the repository you can introduce changes to it as per your wish.
2. After changing it you have to stage the file and then commit it.
3. After committing changes push it to your remote repository.

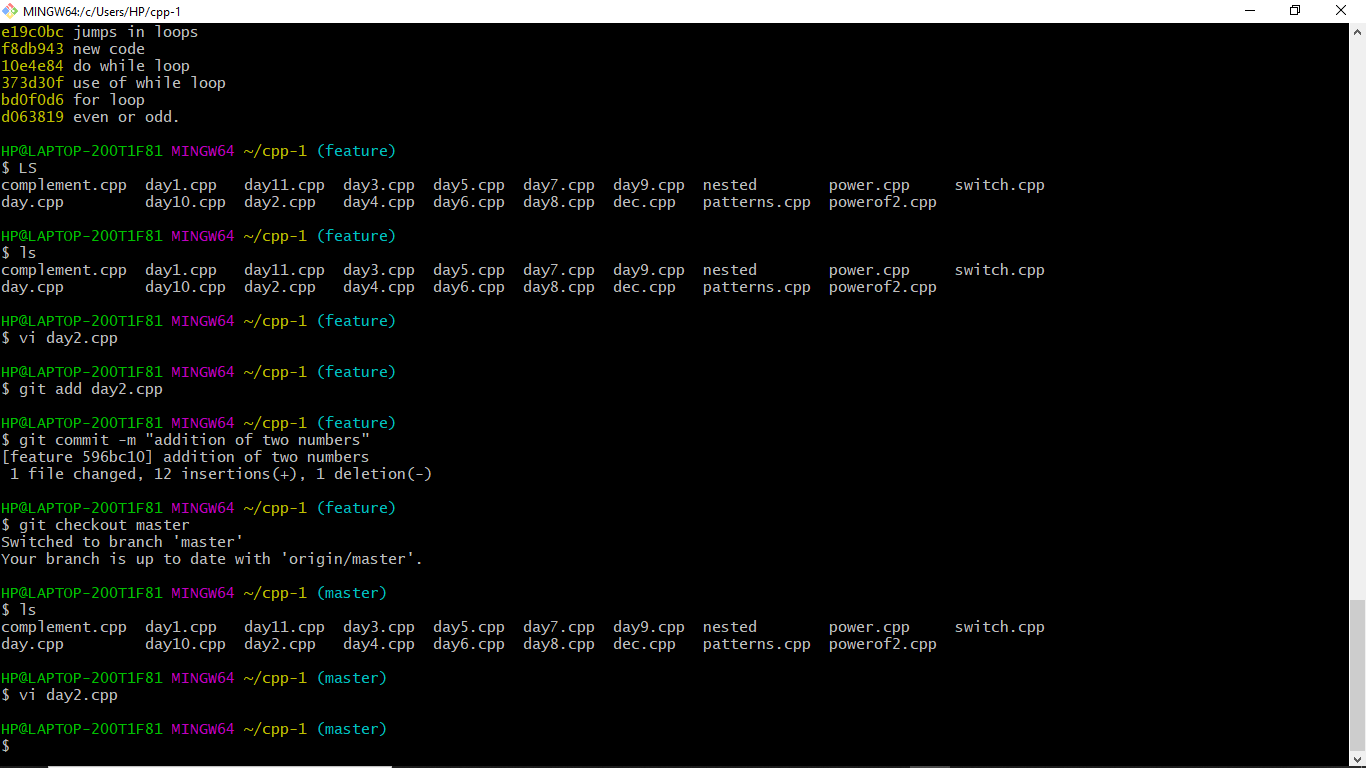


*MERGE AND RESOLVE CONFLICTS CREATED DUE TO OWN ACTIVITY AND COLLABORATORS ACTIVITY*

Merging and conflicts are a common part of the Git experience. Conflicts generally arise when two people have changed the same lines in a file, or if one developer deleted a file while another developer was modifying it. In these cases, Git cannot automatically determine what is correct. Conflicts only affect the developer conducting the merge, the rest of the team is unaware of the conflict. Git will mark the file as being conflicted and halt the merging process. It is then the developers' responsibility to resolve the conflict.

1.To understand the merging concept of branches, create a branch named “feature” in your repository.

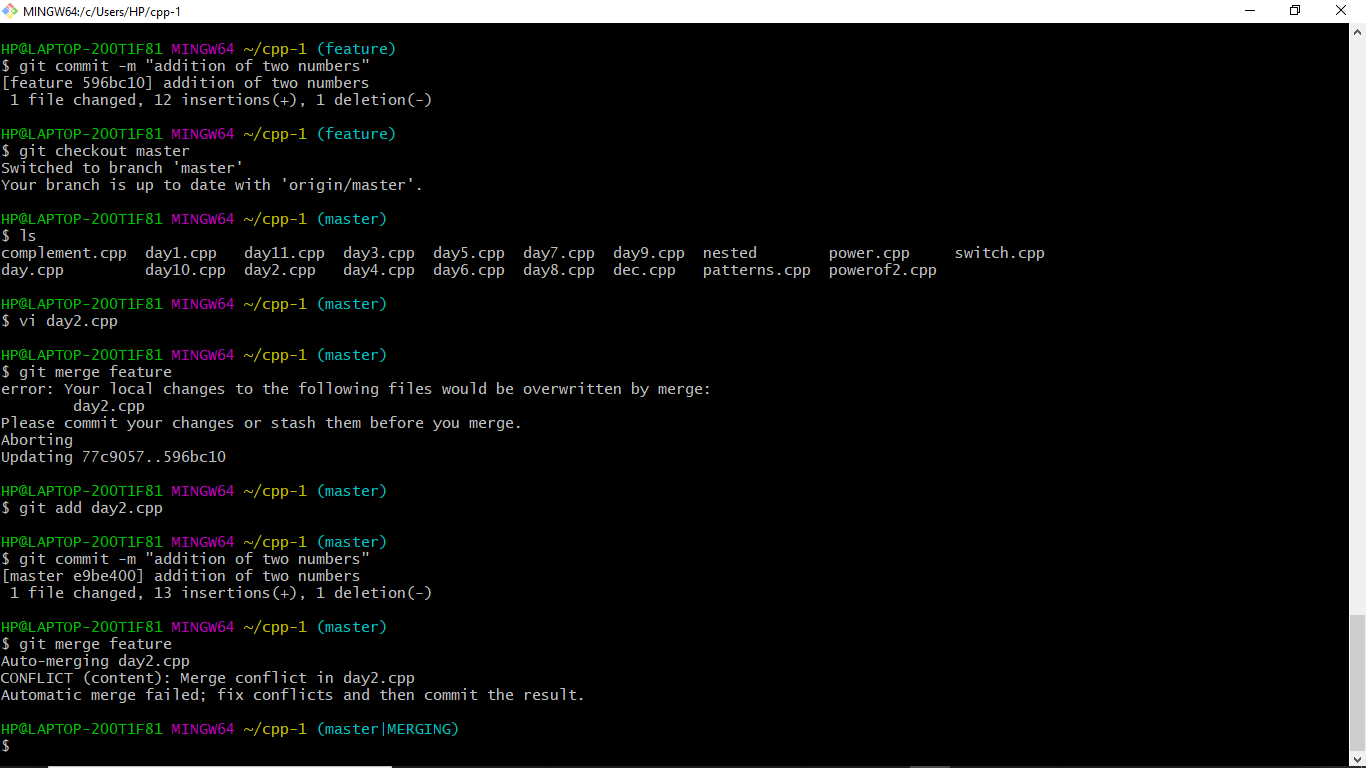


2.Here, there is a file called ‘day2.cpp’. Make changes to it, add and commit them.

3.Similarly, change the same lines of day2.cpp file in the master branch.

4.If you are not already on the branch that you want the other one to merged in (in this example master branch), then switch to it.

5.Using the command try merging feature branch into master branch using the “git merge <branch name>”

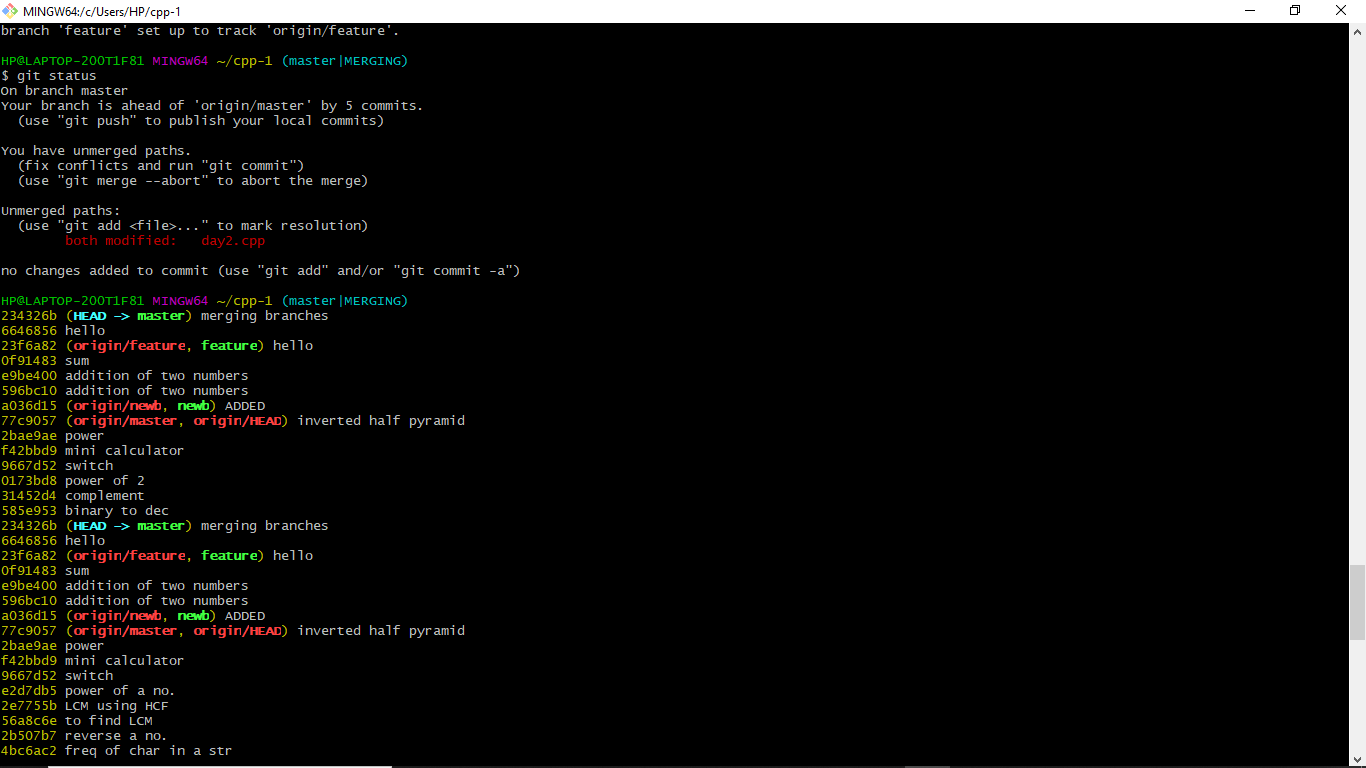






6.Auto merging fails and conflict arises. In order to resolve it we make use of the mergetool by running the command “git mergetool”. The mergetool editor will open.

7.Make changes as per requirement in order to resolve the conflicts and exit the editor.

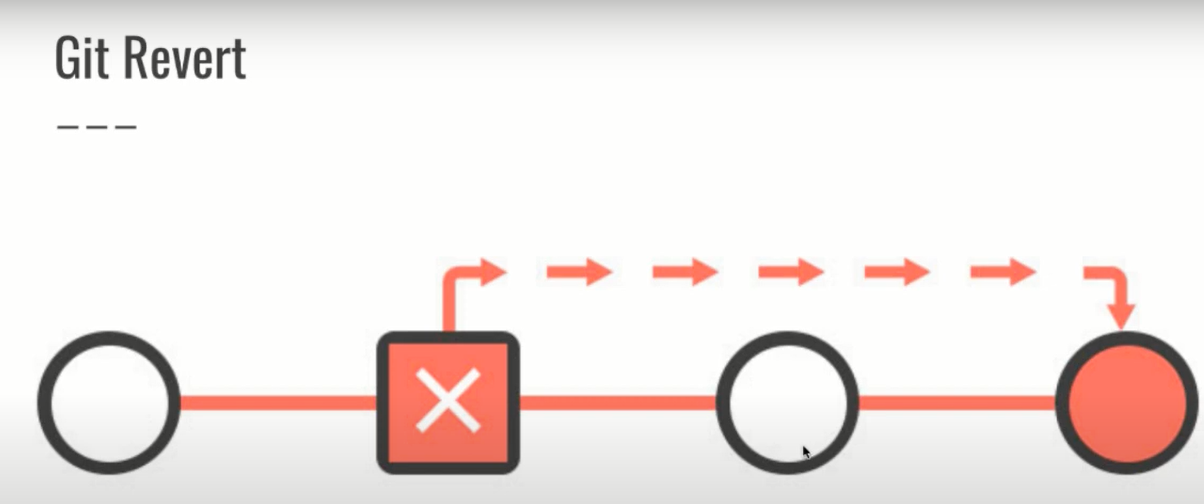


*RESET AND REVERT*

While Working with Git in certain situations we want to undo changes in the working area or index area, sometimes remove commits locally or remotely and we need to reverse those changes. We can do it by using the git reset, git revert, git checkout commands.

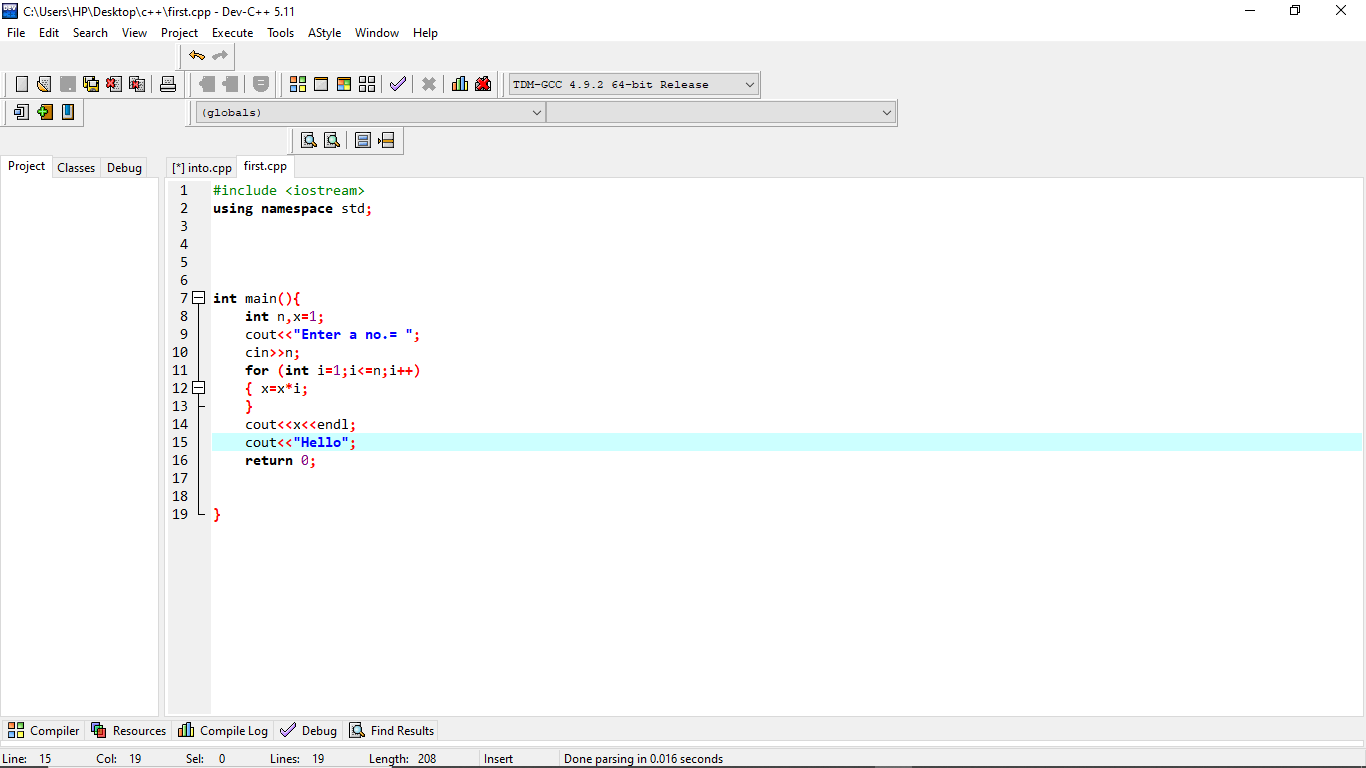
**REVERT**-

git revert is used to remove the commits from the remote repository. git revert removes the commit that we have done but adds one more commit which tells us that the revert has been done.

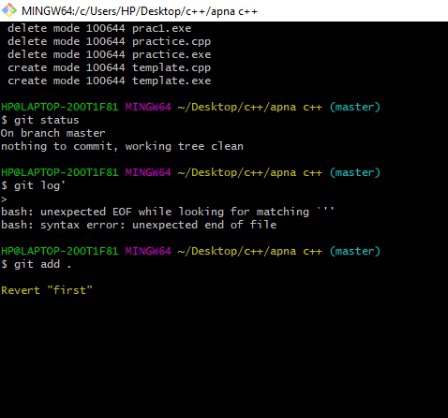


Let’s see how to revert a commit, say haved pushed a unwanted commit from your local and now we will revert it.

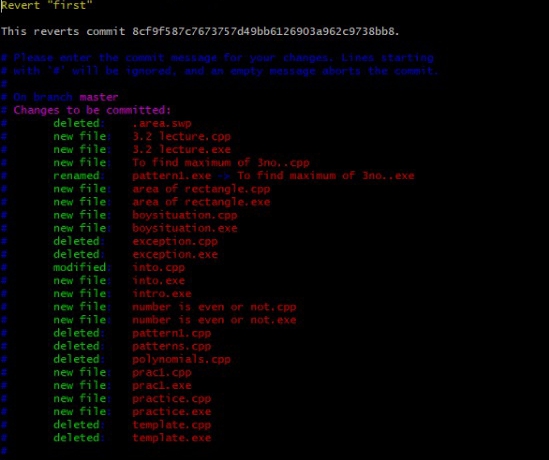
The basic advantage of reverting a commit is that it is not permanentatly deleted.



Now to revert the changes made in the commit run the “git revert <commit id>” command.



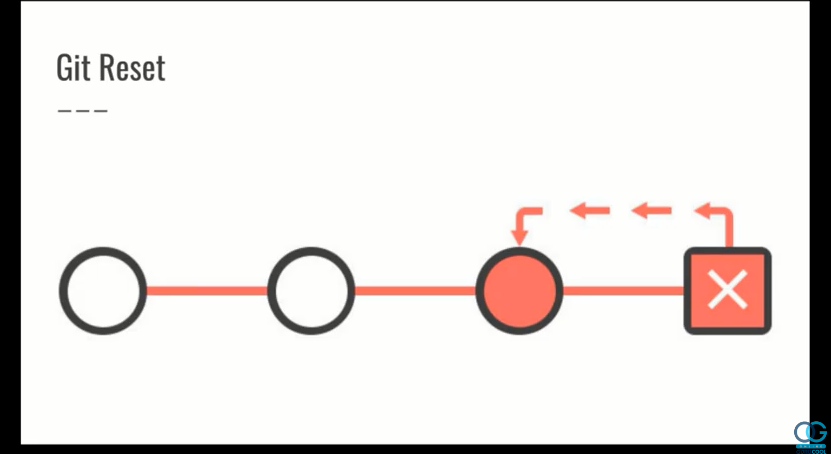


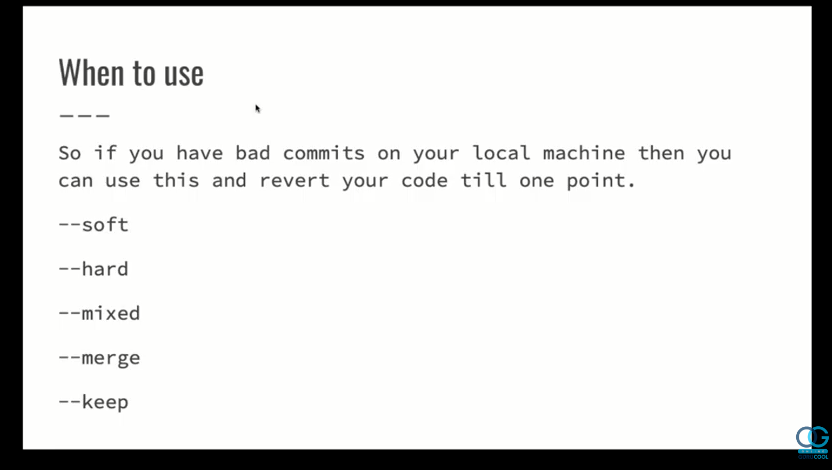


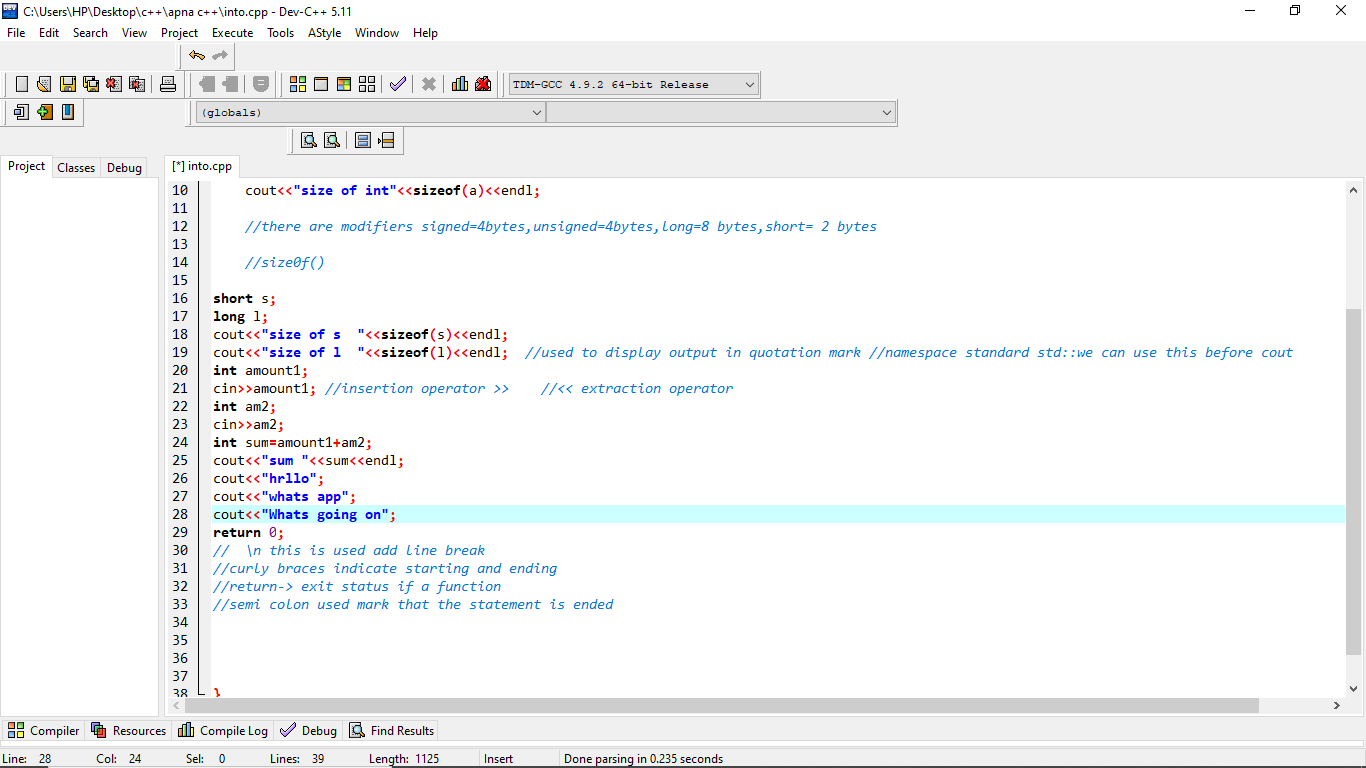
**RESET**-

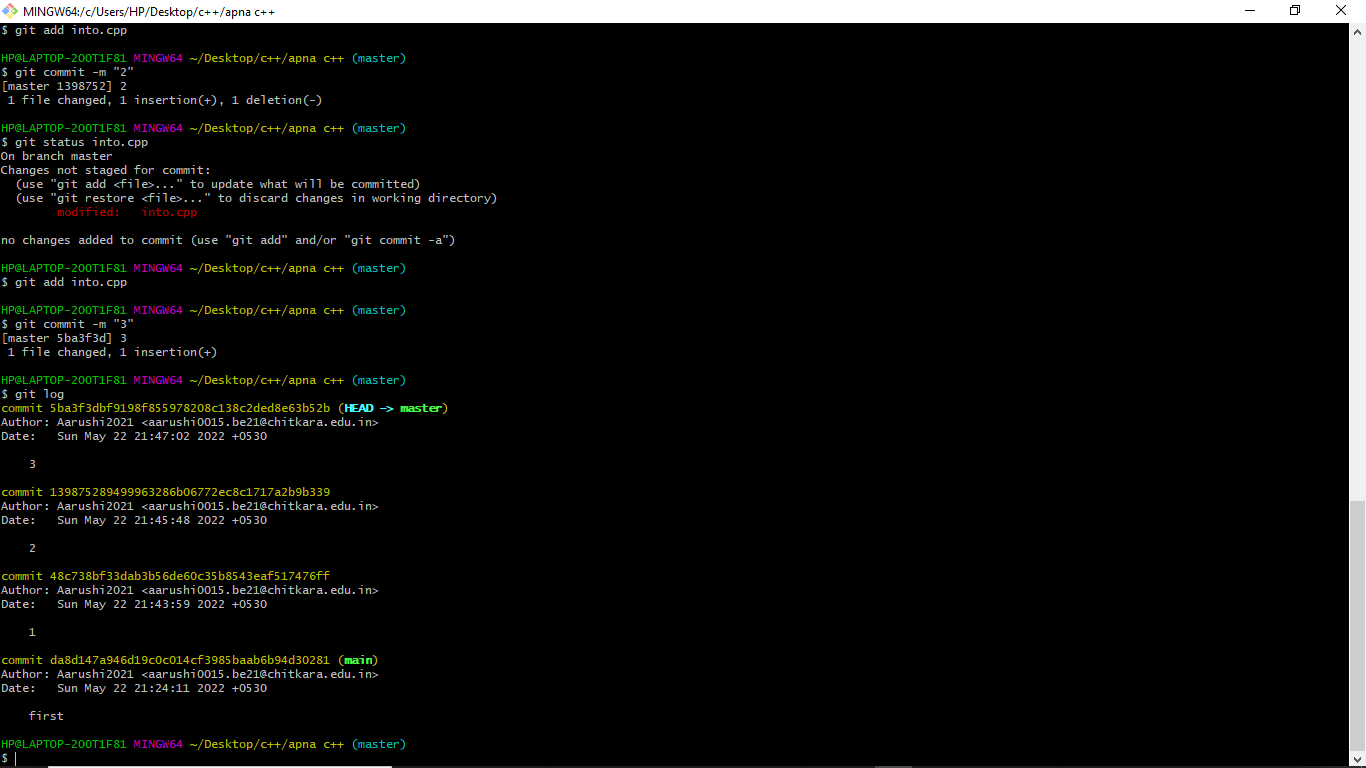
git reset is used when we want to unstage a file and bring our changes back to the working directory. Git reset can also be used to remove commits from the local repository.

Suppose we make edits to a file, stage it and commit it







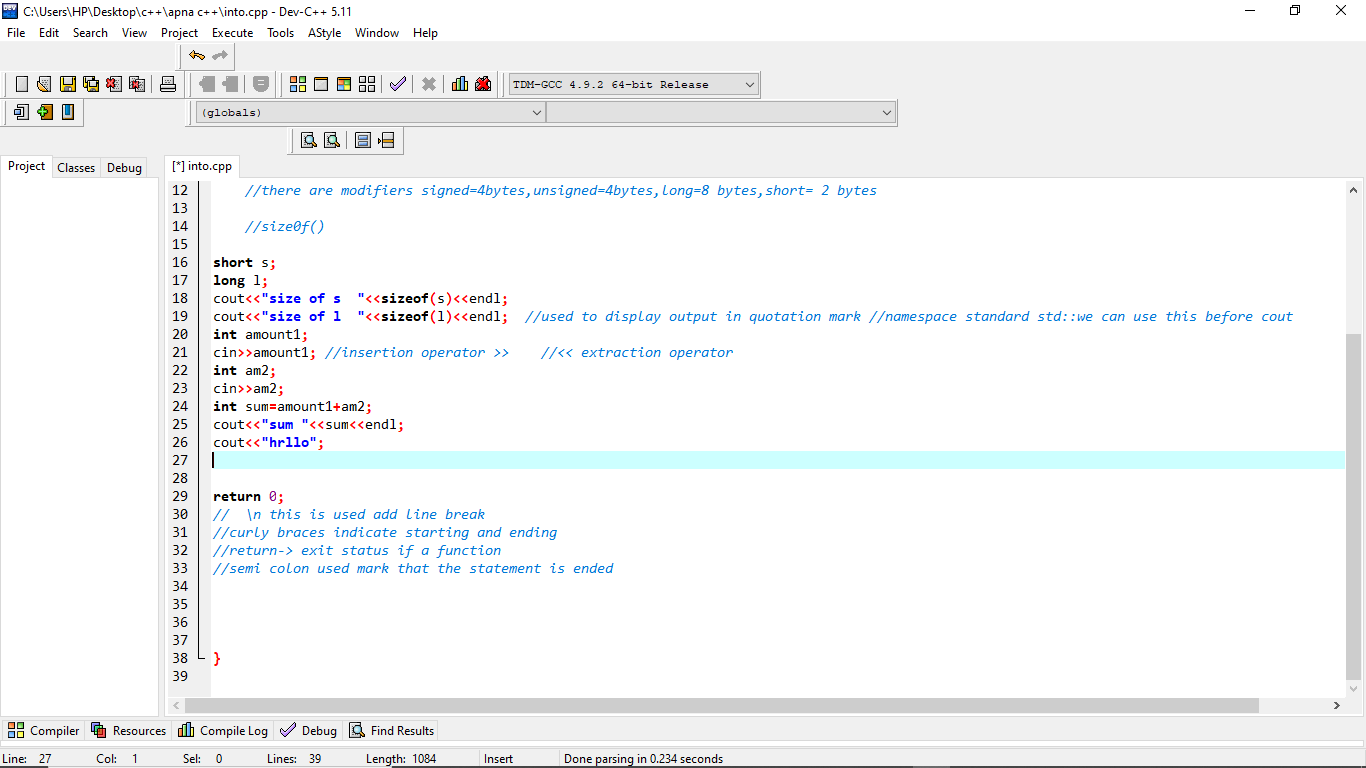


In order to reset the changes made in the recent commit, run the “git reset --hard HEAD~1” command.

Or a command git “reset commit no.”



The HEAD returns to the previous commit and the changes made are reset.





TASK 2

|  |  |
| --- | --- |
| **S. No.** | **Title** |
| 1 | Introduction |
| 2 | Creating a repo |
| 3 | Open and Close Pull request |
| 4 | Fork ,clone and creating pull request |

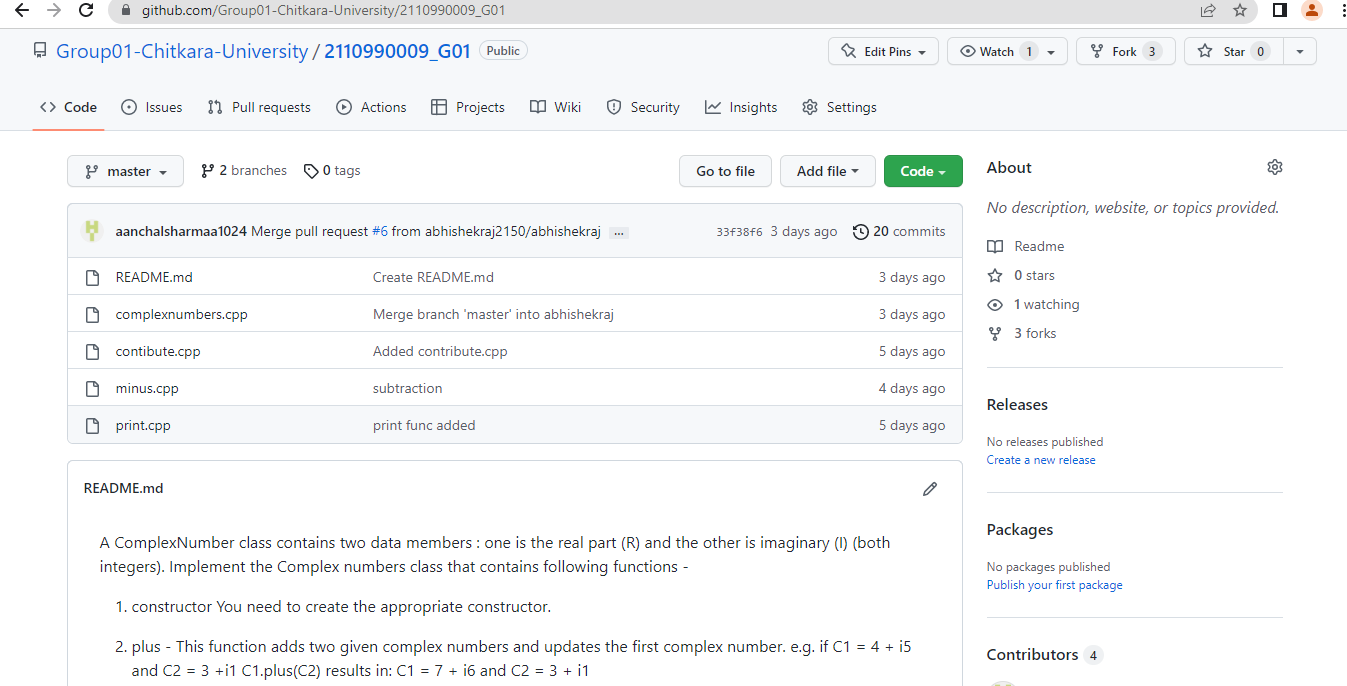
INTRODUCTION:

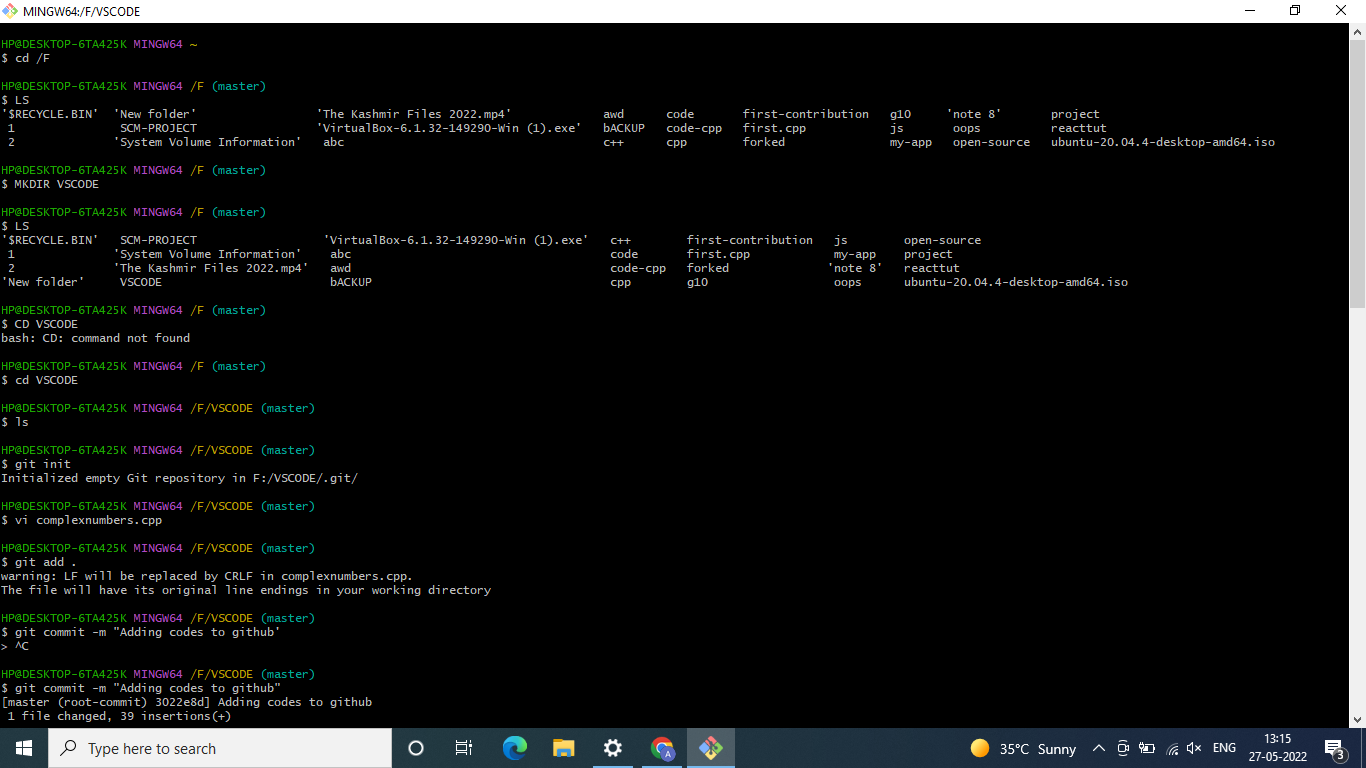
This task is performed in the group of four. Each one of us made it possible to work on this project as if we are doing an open source contribution.

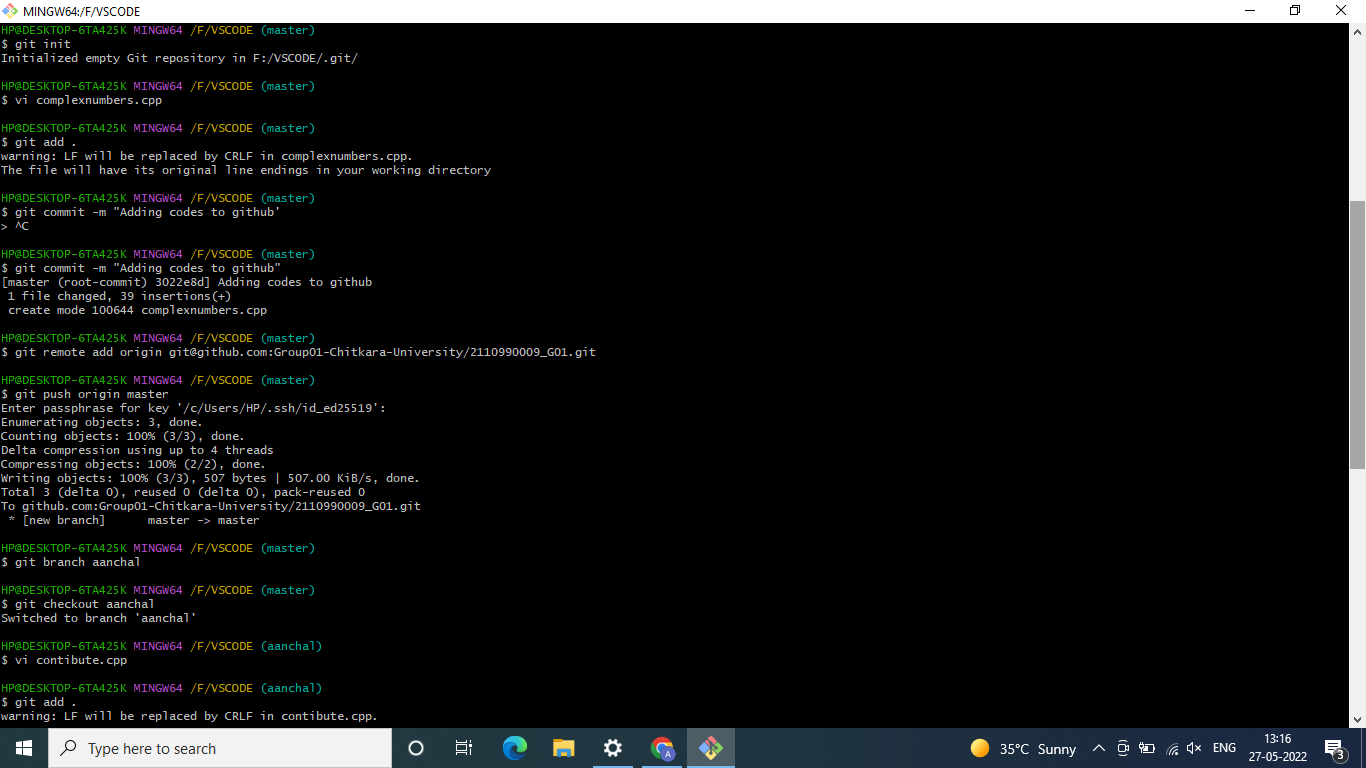
Each one of us create his/her repo and rest of the three contributers in the repo , firstly forked that repo and then clone it in our local machine and then make a new branch and made some changes in the existing file in master branch in the repo and then push it from your local system.

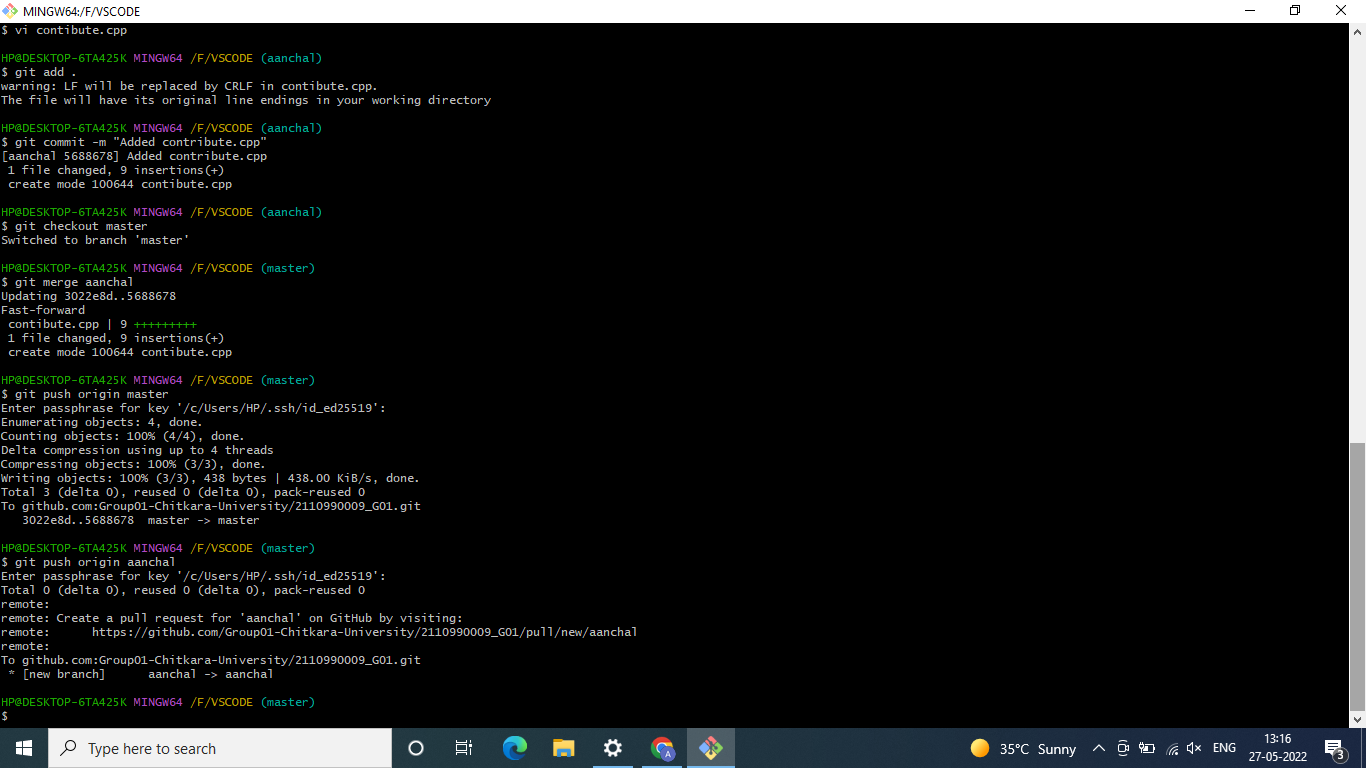
And finally make pull request to the owner of the repo in whose repo we want to make the changes.

CREATING A REPO :







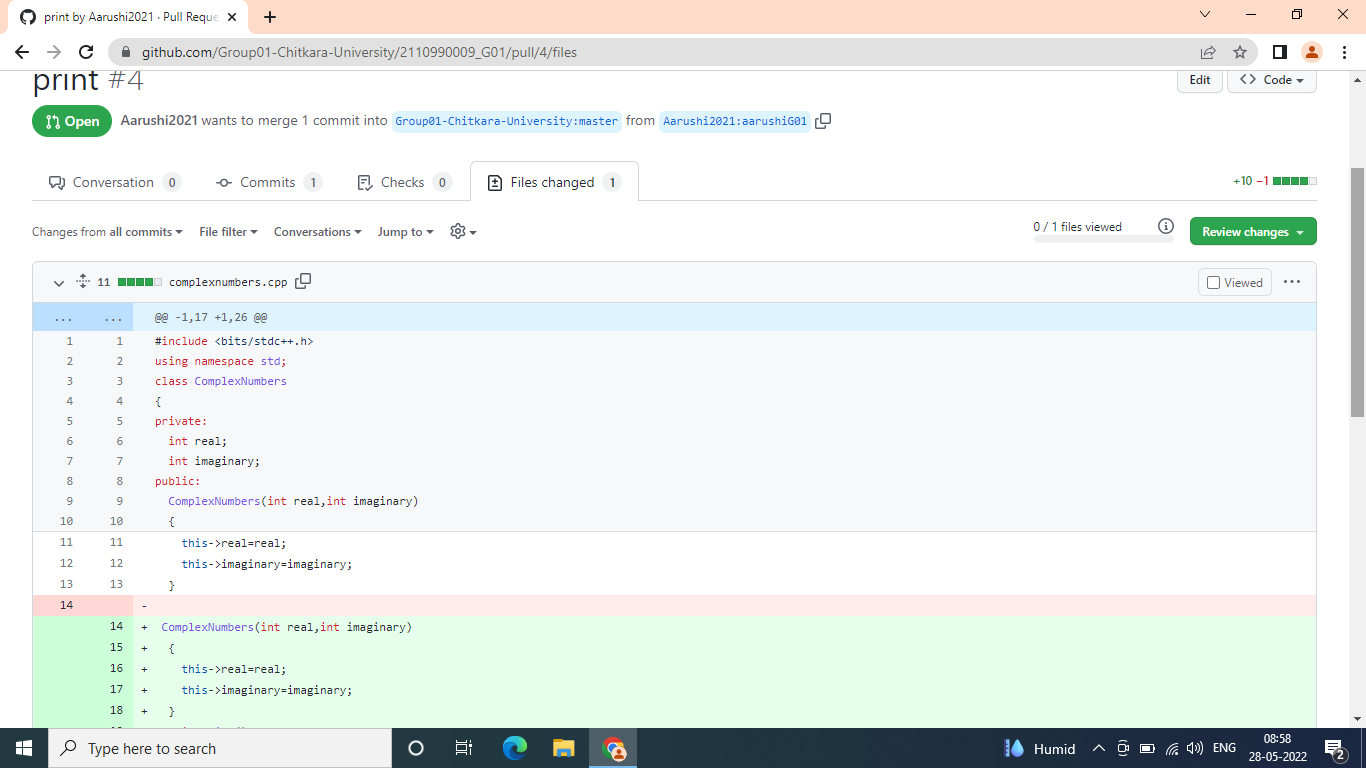


OPEN AND CLOSE PULL REQUEST:

Overview of the pull request sent by the contributers.

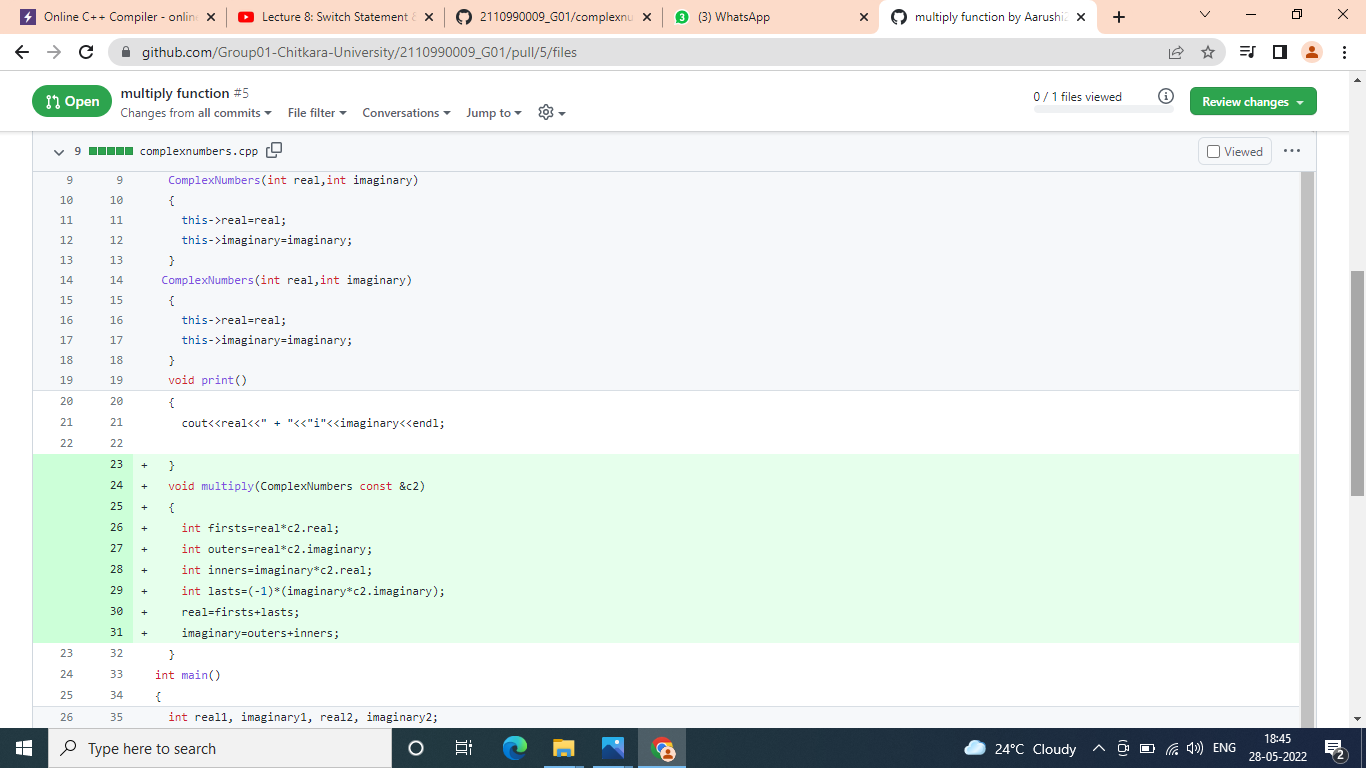
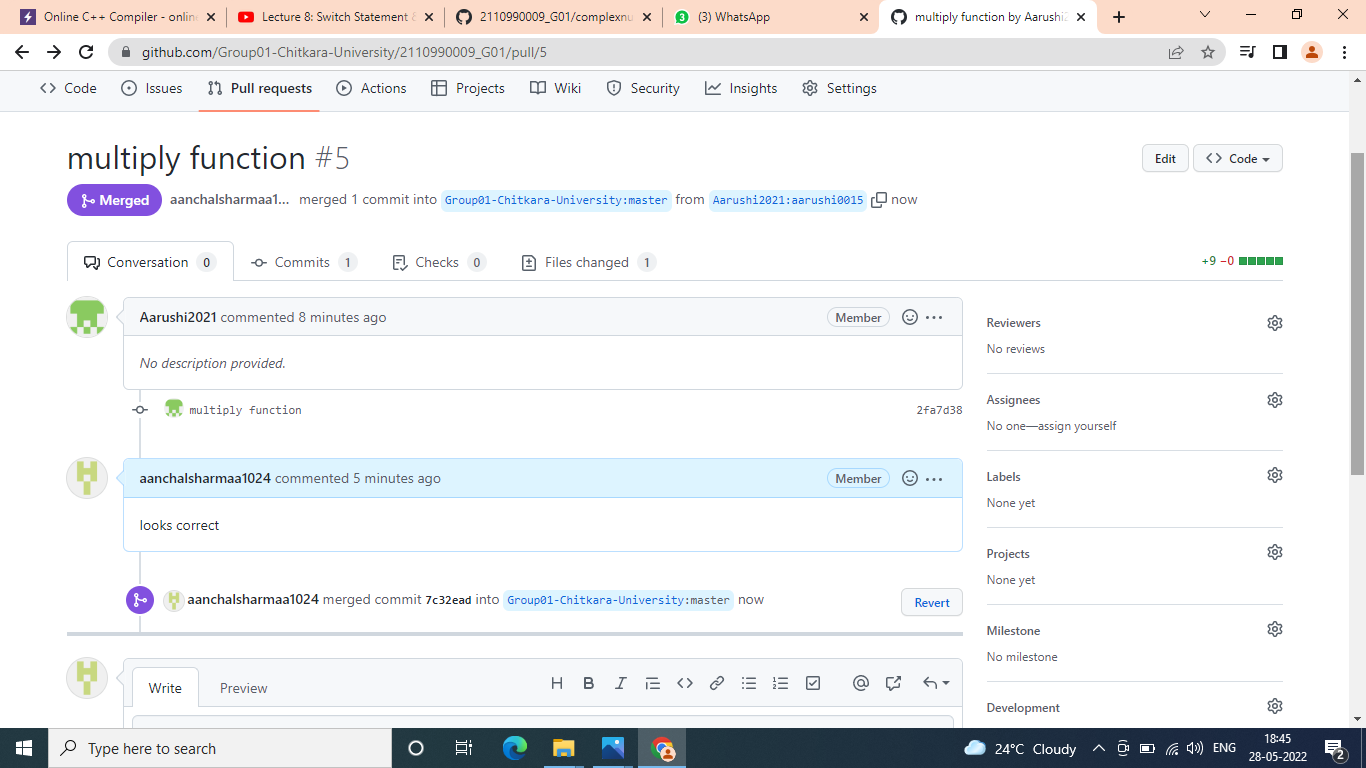
And if in case there is a confict in merging , solved it by mergetool in the git bash.





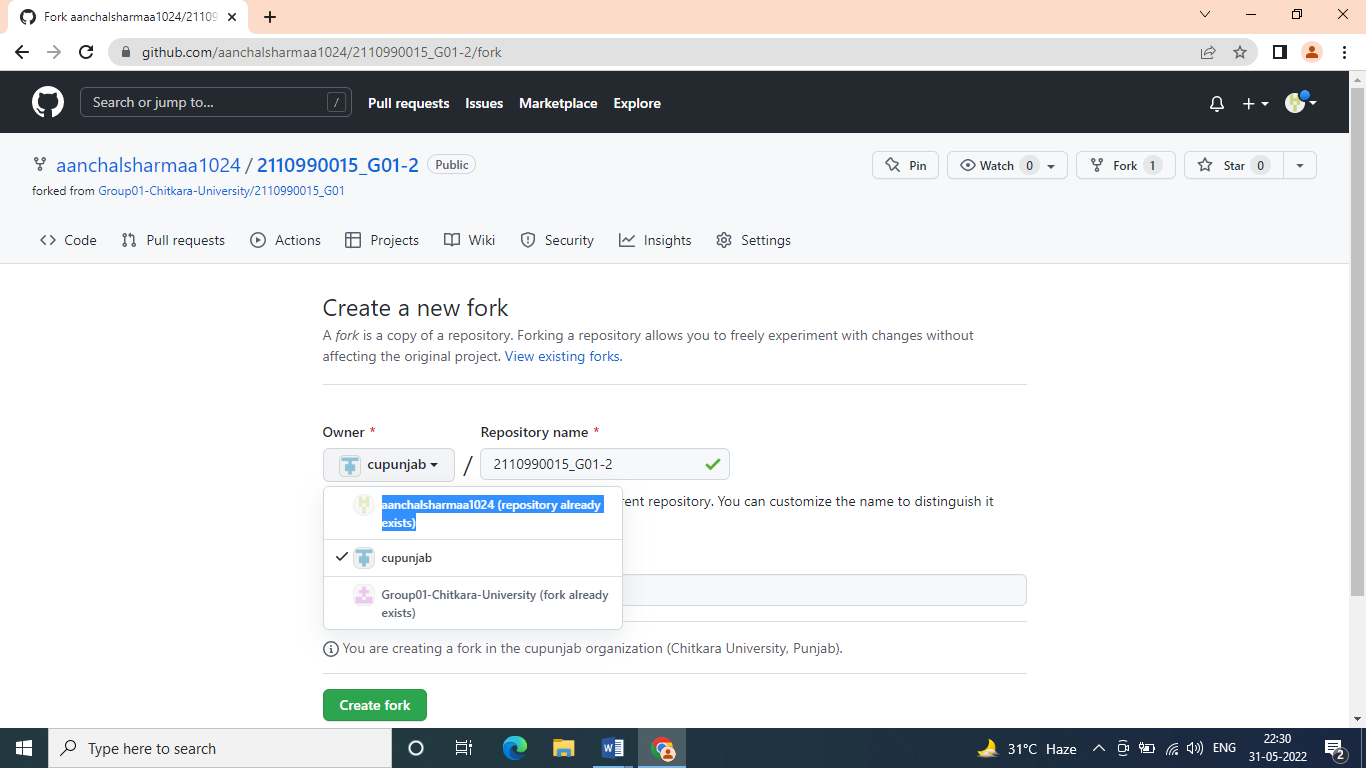




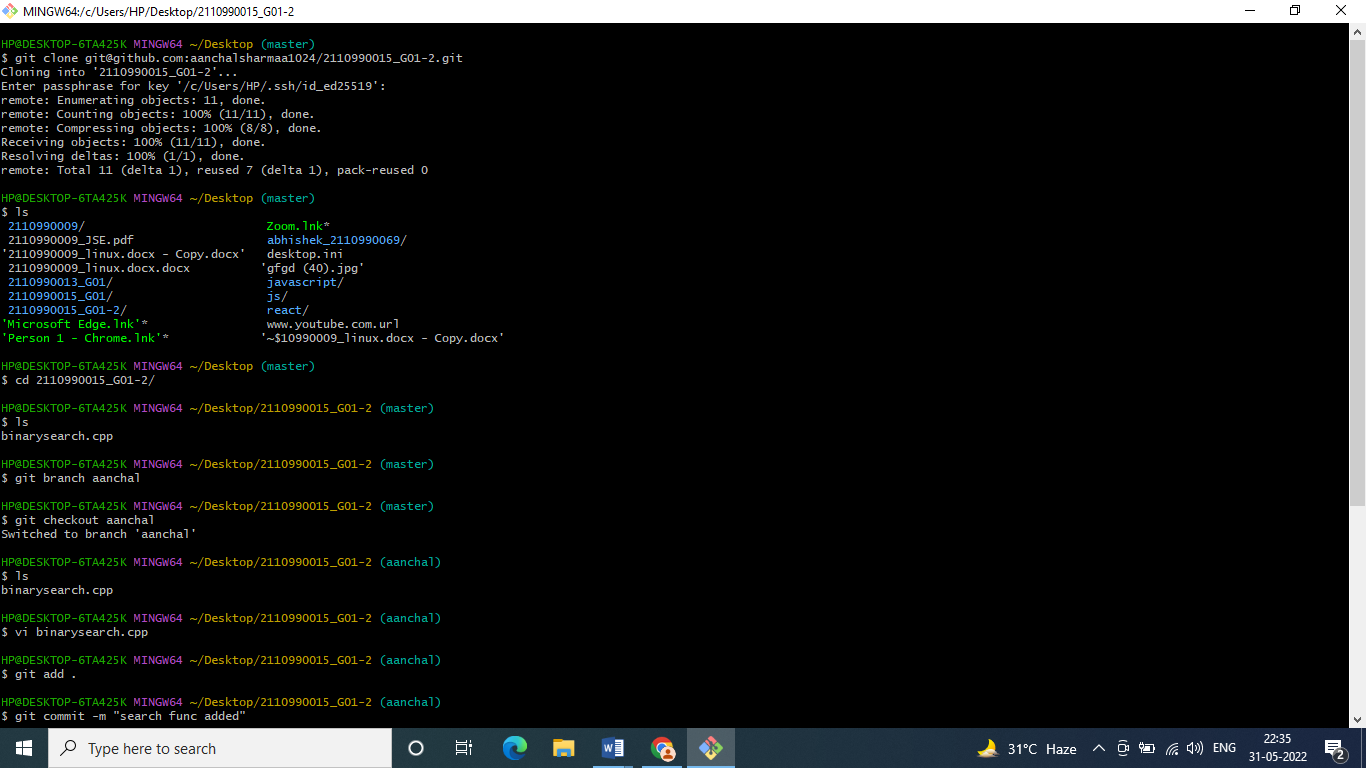


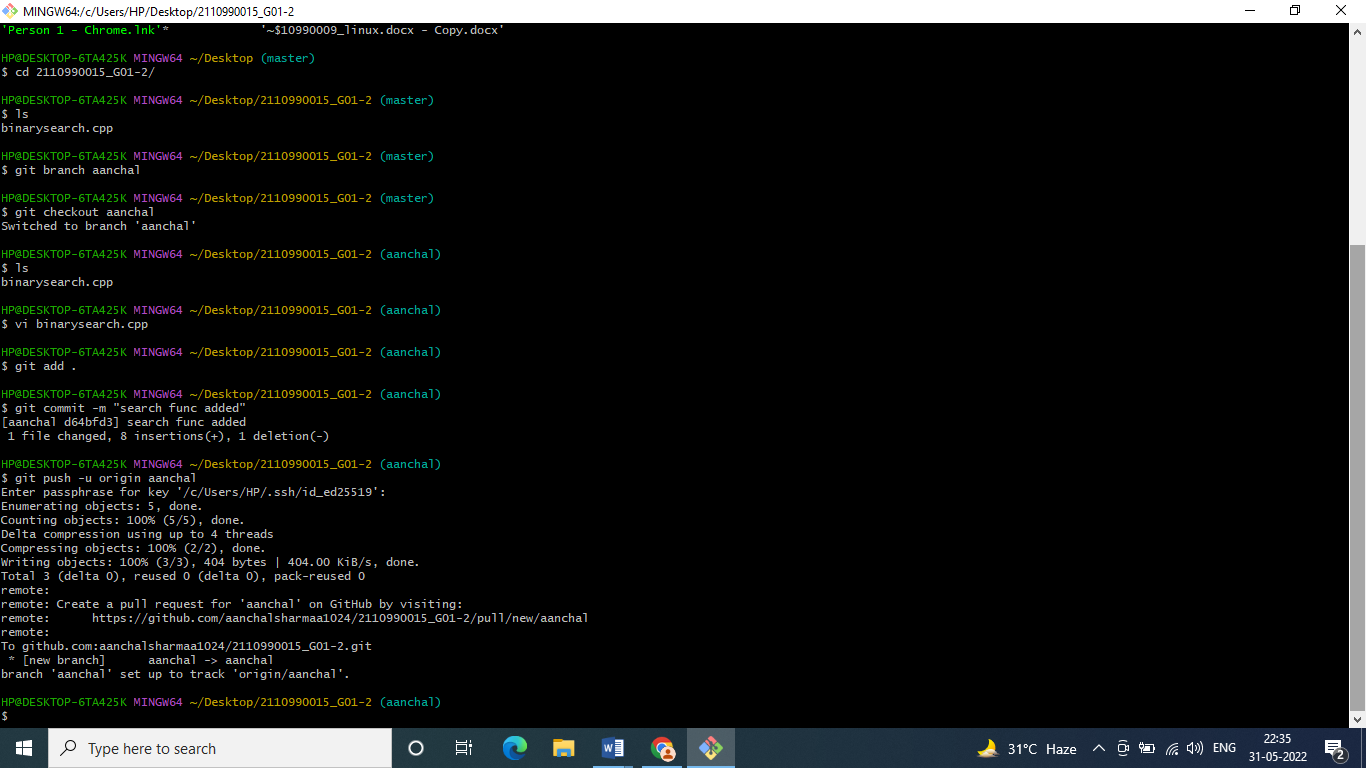
FORK AND CLONING THE REPO’S :

1. Create fork by selecting yourself as a owner , and click on create fork.



1. Cloning in your remote system and making a branch to add your changes . and finally pulling a pull request .





1. Create a pull request by clicking on compare and pull request.

