# SOURCE CODE MANAGEMENT FILE

Subject: Source Code Management

Code:CS181

Submitted By:

NAME: Akshi Mital

ROLL NO: 2210991232

GROUP:02

|  |  |  |
| --- | --- | --- |
| Sno. | TITLE | Page no. |
| 1. | Setting up Git Client |  |
| 2. | Setting up GitHub Account |  |
| 3. | Generate logs |  |
| 4. | Create and visualize branches |  |
| 5. | Git life cycle description |  |
| 6. | Add collaborations on GitHub Repo |  |
| 7. | Fork and Commit |  |
| 8. | Merge and Resolve conflicts created due to own activity & collaborators activity |  |
| 9. | Reset and revert |  |

# Experiment-1

**Aim:** Setting up of Git Client

**Theory:**

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. 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.

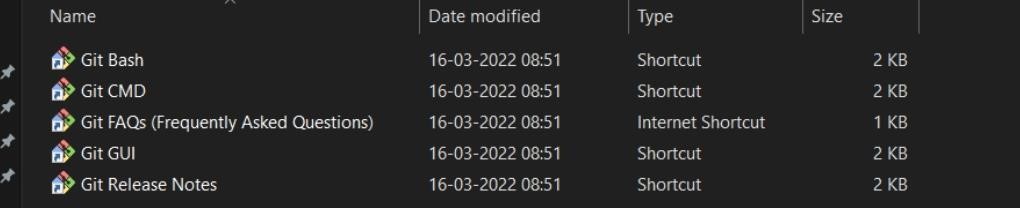
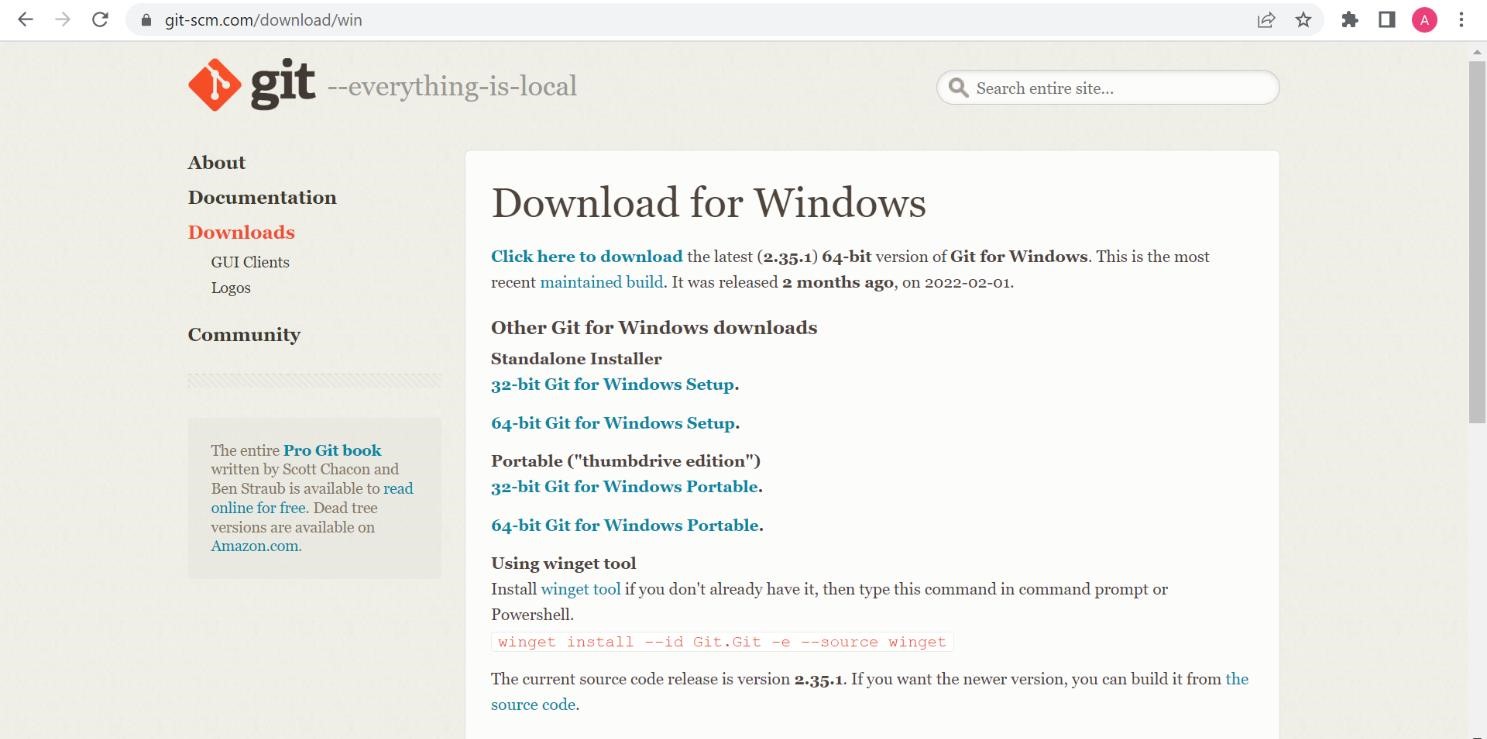
**Procedure:** We can install Git on Windows, using the most official build which is available for download on the GIT’s official website or by just typing (scm git) on any search engine. We can go on <https://git-scm.com/download/win>and can select the platform and bit-version to download. And after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

**Snapshots of download:**

Opted for “64

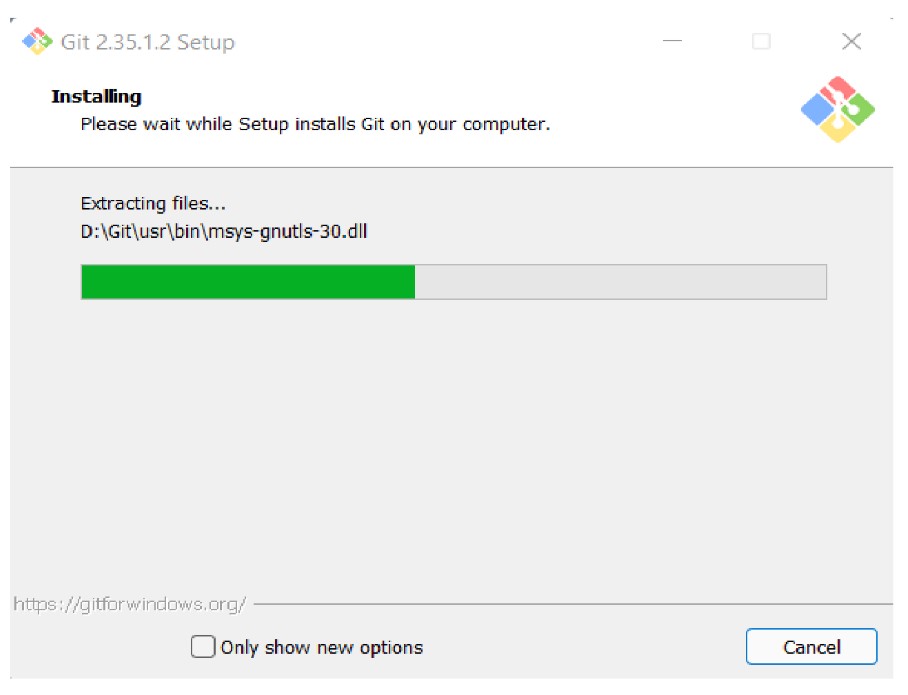
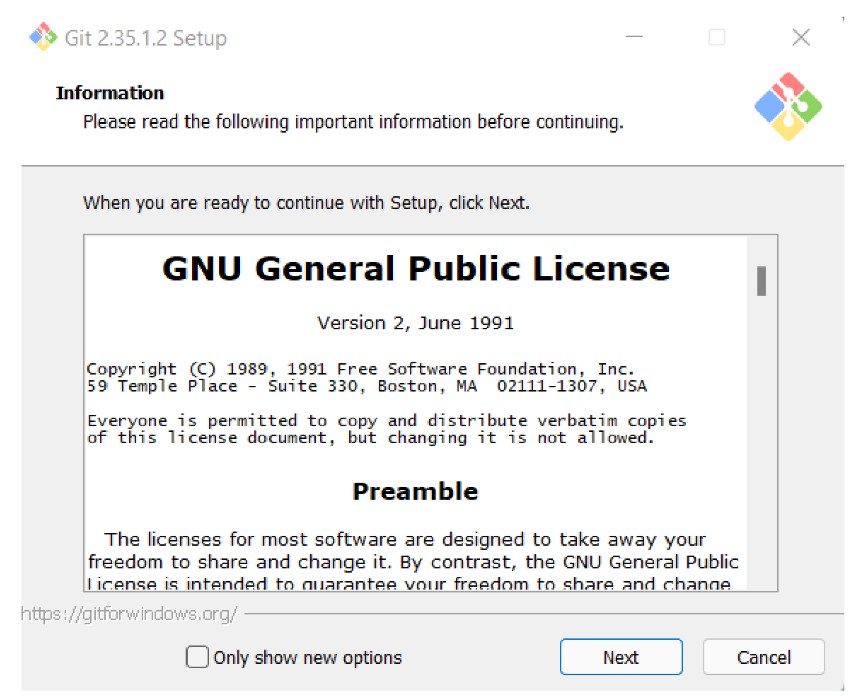
-

bit Git for Windows Setup”

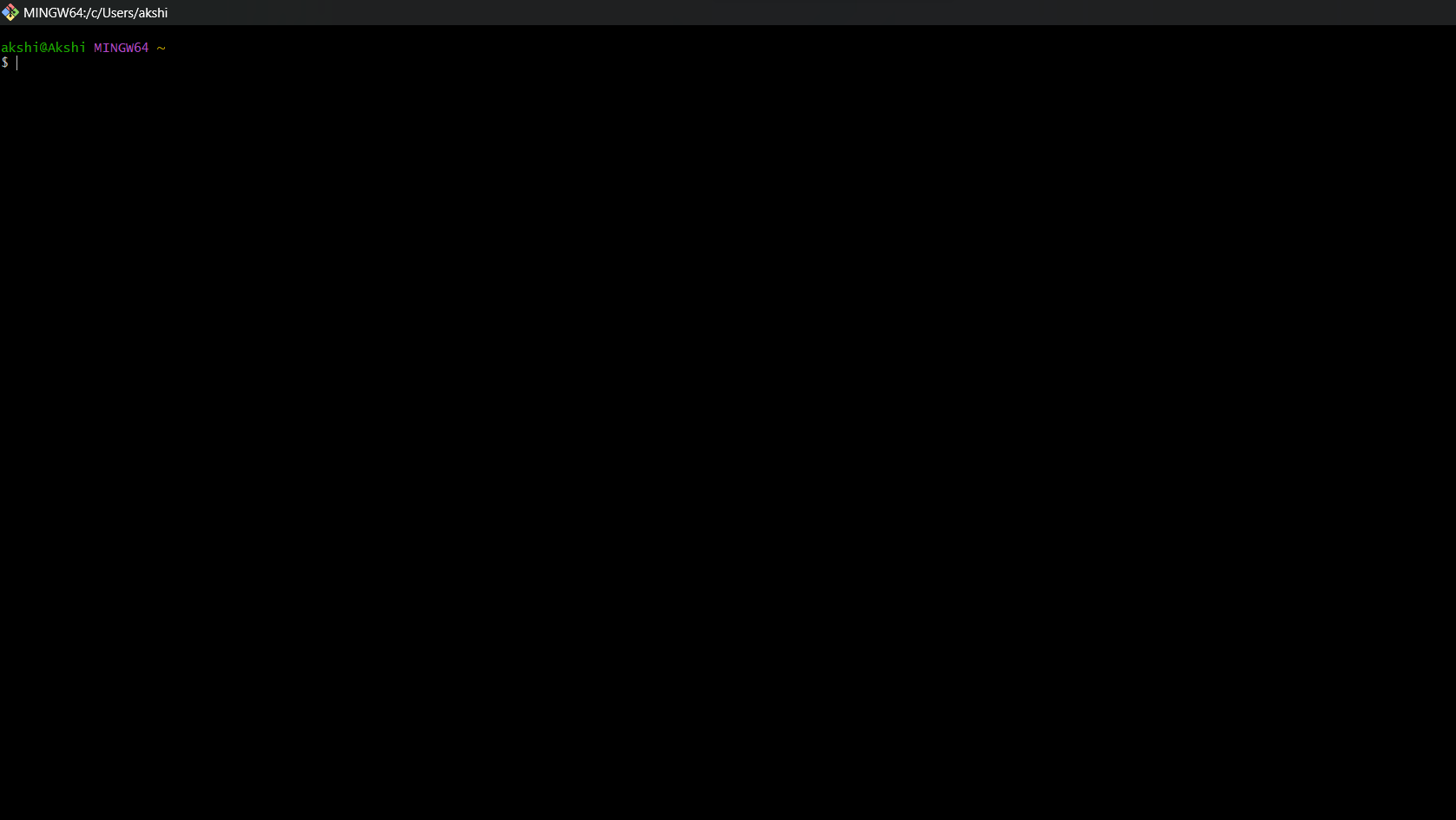


## Git and its files in downloads

Git Setup



Git Installation



GIT BASH LAUNCHED

# Experiment 2

**Aim:** Setting up GitHub Account

**Theory:**

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

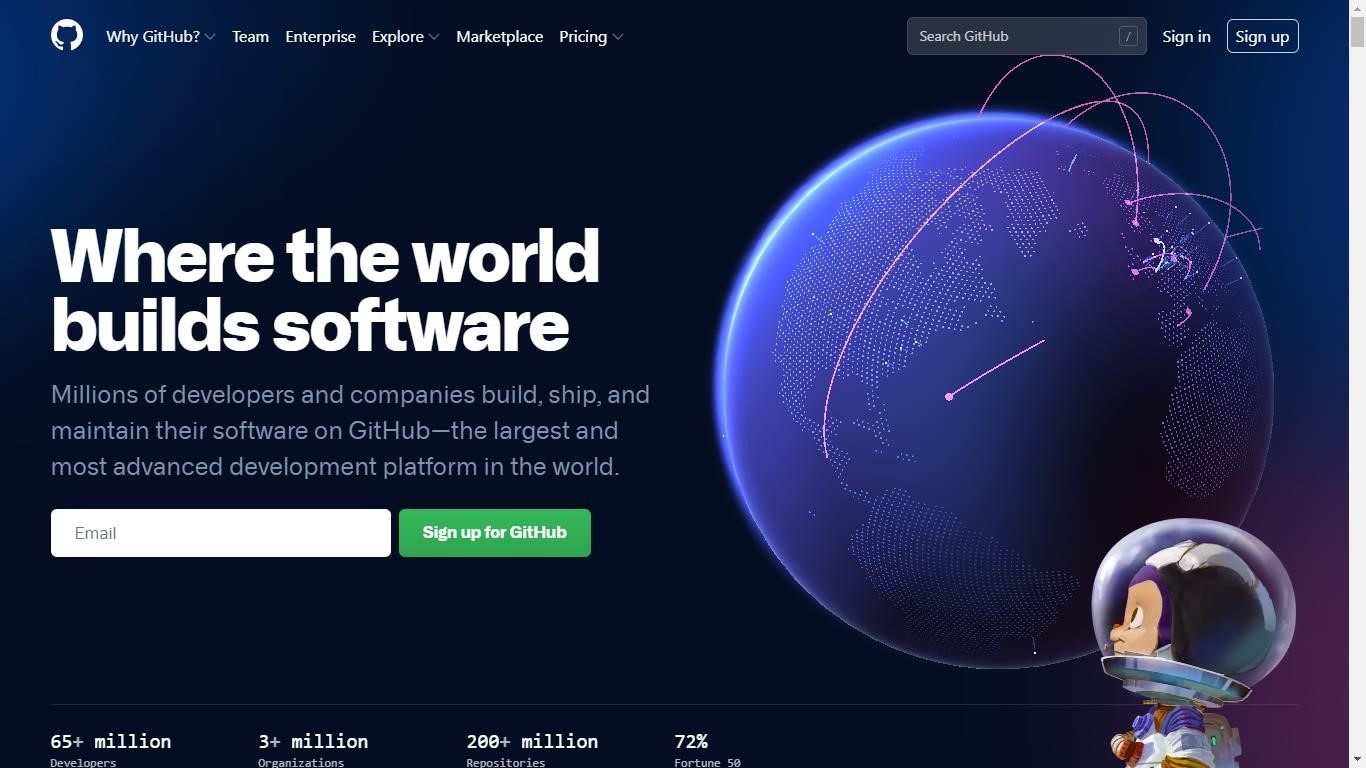
Advantages of GitHub: GitHub has a user-friendly interface and is easy to use.We can connect the git-hub and git but using some commands shown below in figure 001. Without GitHub we cannot use Git because it generally requires a host and if we are working for a project, we need to share it will our team 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:**

To make an account on GitHub, we search for GitHub on our browser or visit [https://github.com/signup.](https://github.com/signup) Then, we will enter our mail ID and create a username and password for a GitHub account.

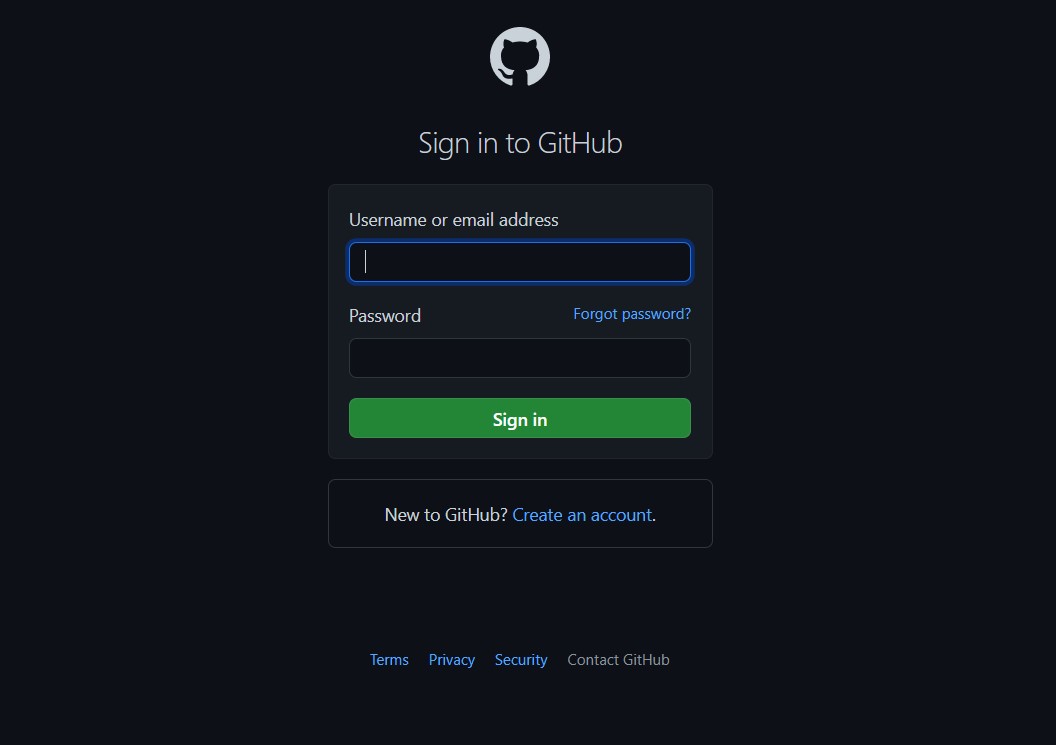
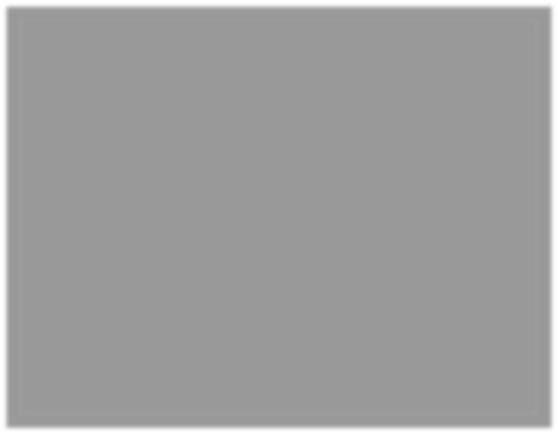
**Snapshots**

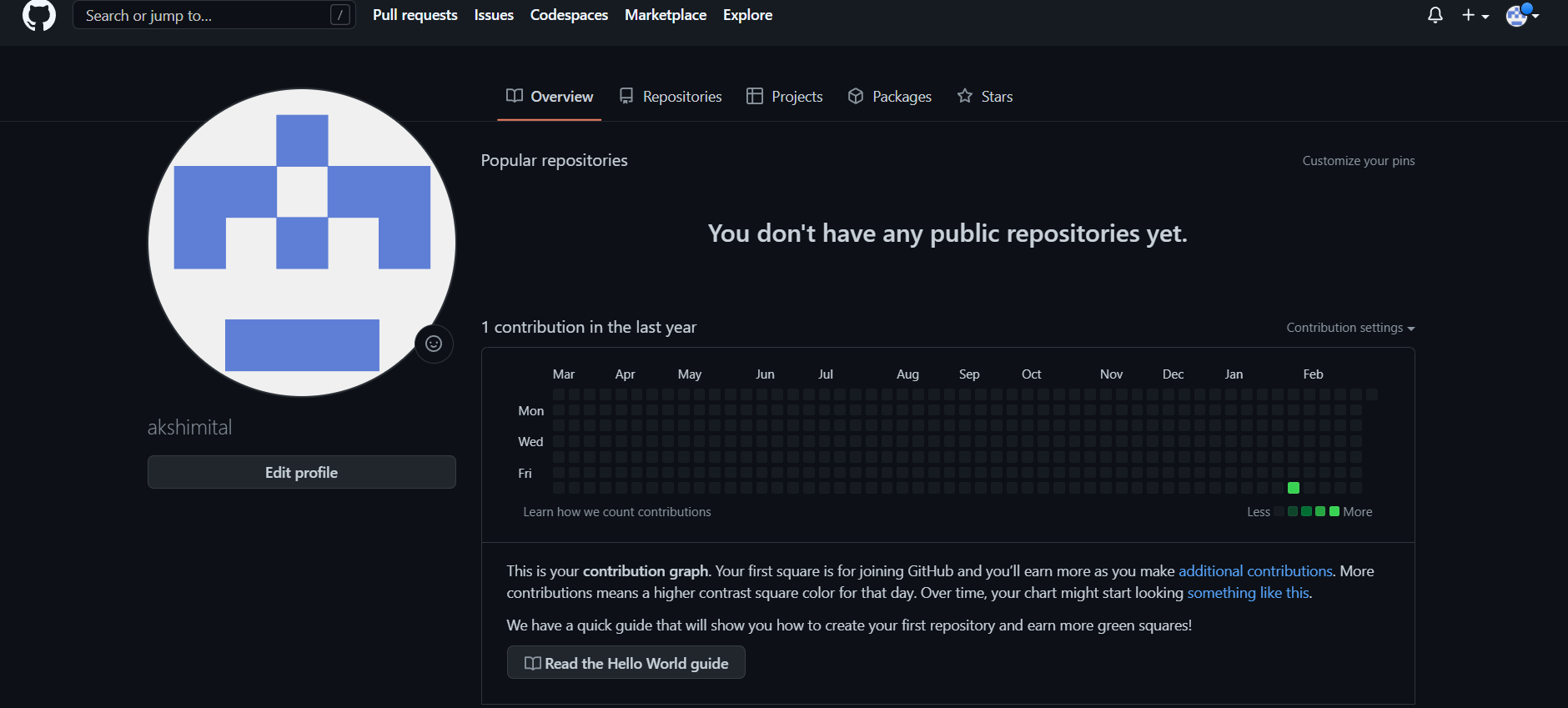
**–**



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

GitHub Login





GitHub Interface

# Experiment 3

**Aim:** Program to Generate log

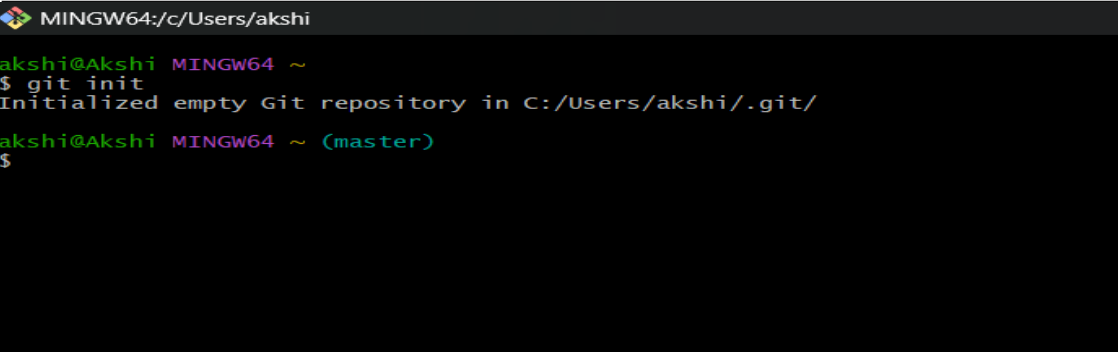
**Theory:**

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. 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.

**Procedure:**

First of all, create a local repository using Git. For this, you have to make a folder in your device, right click and select “Git Bash Here”. This opens the Git terminal. To create a new local repository, use the command “git init” and it creates a folder “.git”.



## GIT INITIALISATION

When we use GIT for the first time, we have to give the user name and email so that if I am going to change in project, it will be visible to all.

For this, we use command:

“git config --global user.name Name”

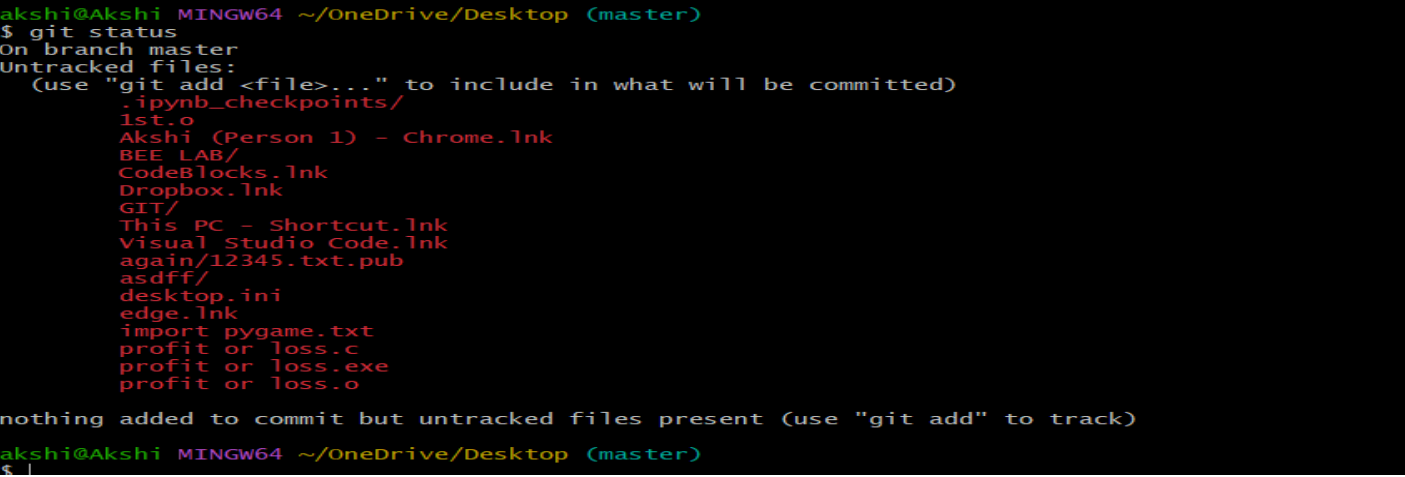
“git config --global user.email email”

For verifying the user’s name and email, we use:

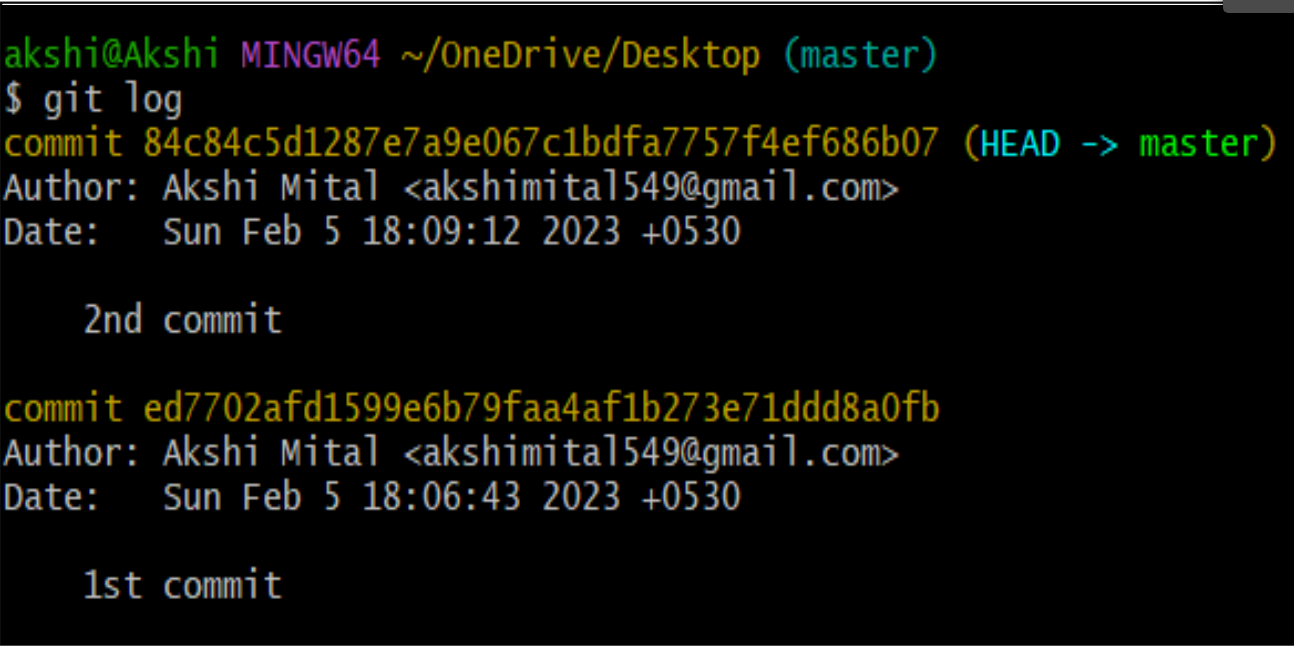
“git config --global user.name”

“git config --global user.email”

GIT STATUS:



GIT LOG:



The git log command displays a record of the commits in a Git repository. By default, The git log command displays a record of the commits in a Git repository. By default,metadata.

# Experiment 4

**Aim:** Create and visualize branches

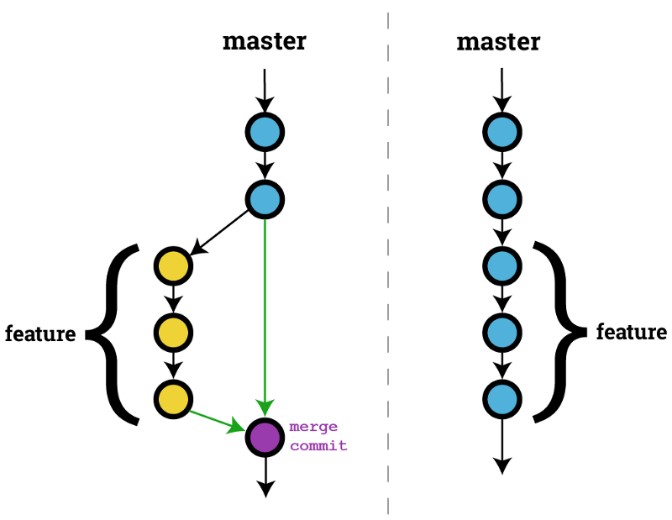
**Theory:**

Branching: A branch in Git is an independent line of work (a pointer to a specific commit). It allows users to create a branch from the original code (master branch) and isolate their work. Branches allow you to work on different parts of a project without impacting the main branch.

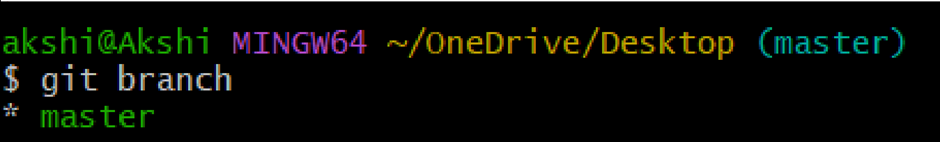
Create branches: The main branch in git is called as master branch. But we can make branches out of this main master branch. All the files present in master can be shown in branch but the file which are created in branch are not shown in master branch. We can also merge both the parent (master) and child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

Syntax:

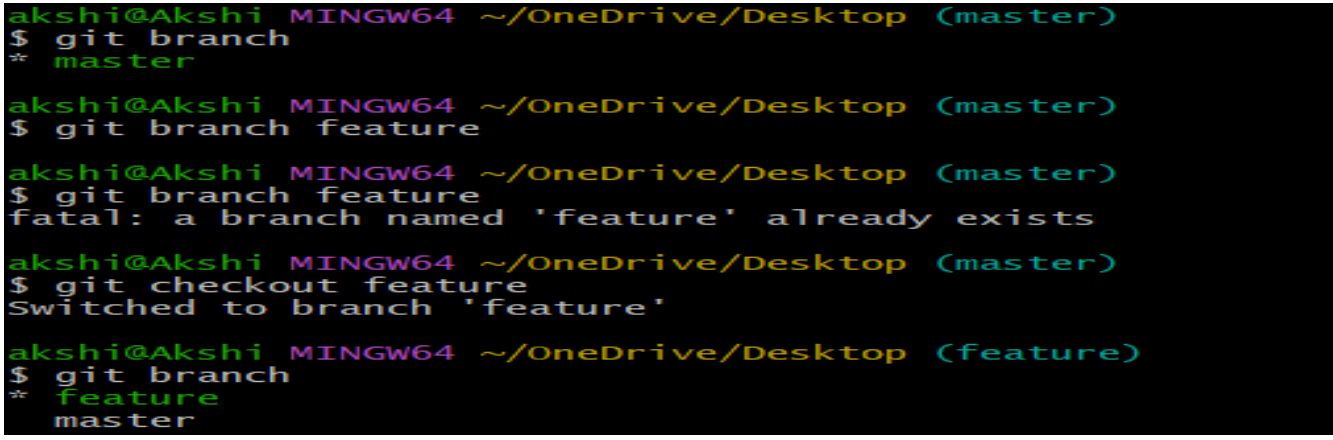
For creating a new branch, git branch name by default is master branch.



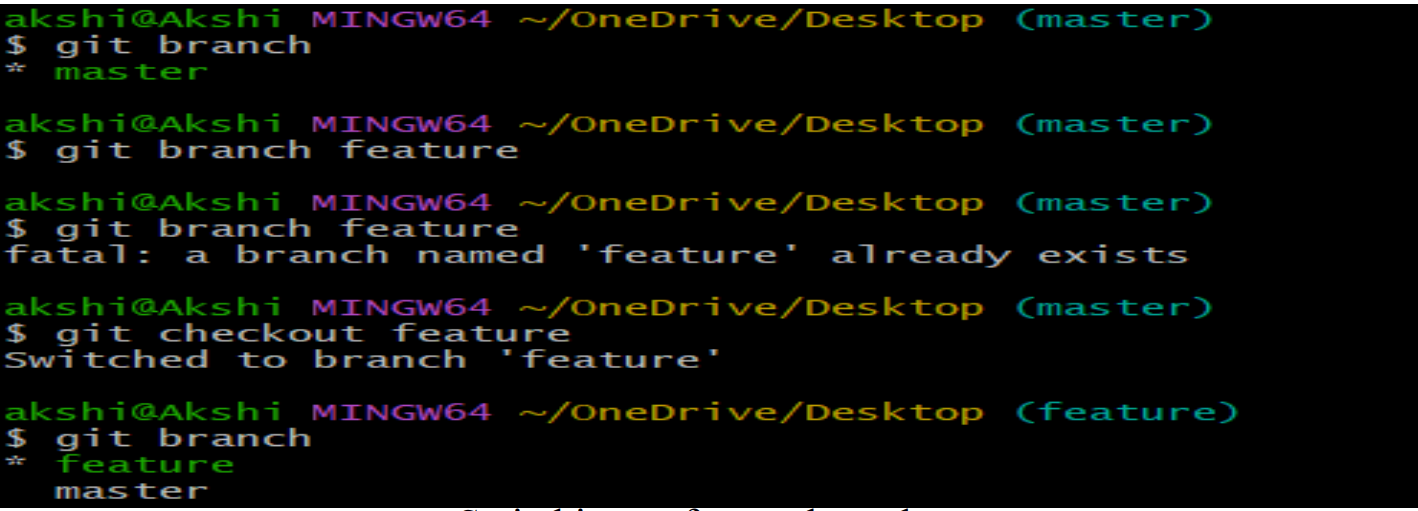
**Snapshots –**



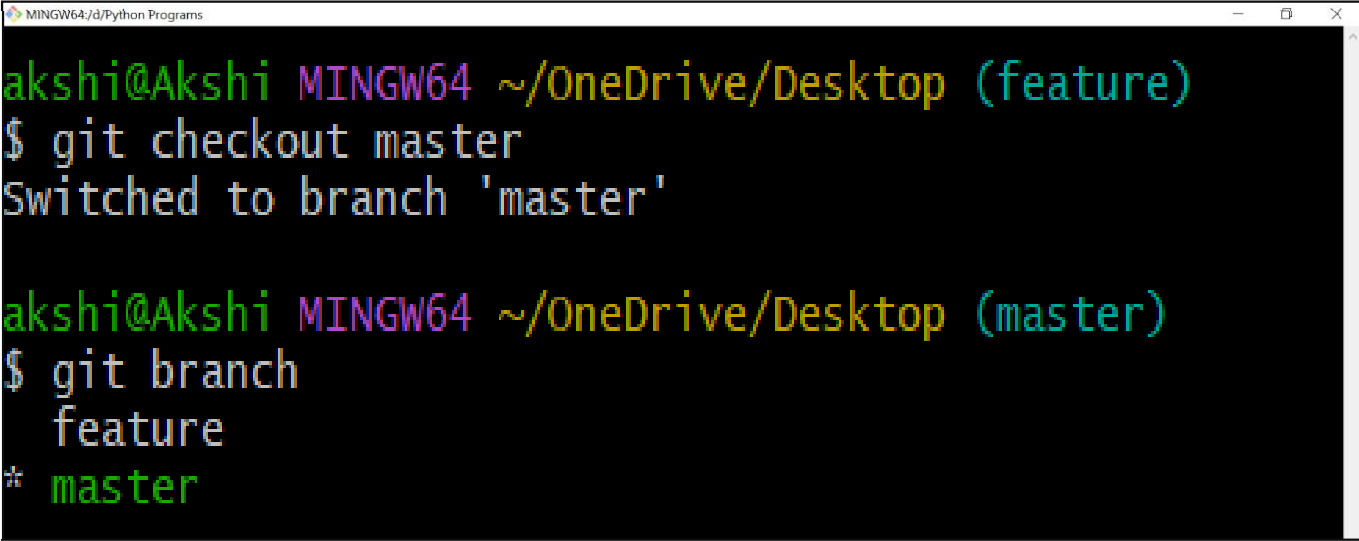
## Default branch is master branch



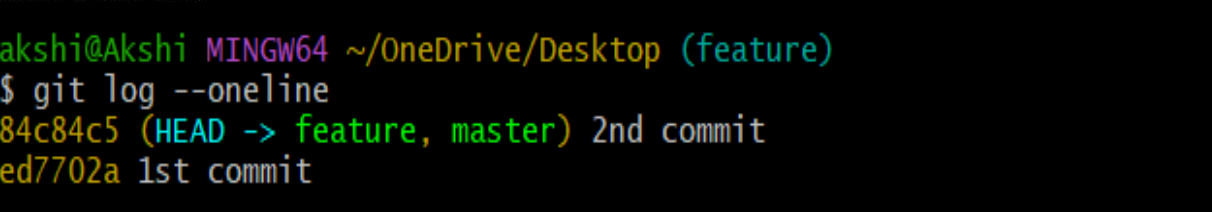
Adding Feature Branch



Switching to Feature Branch



Switching to Master Branch



Checking commits

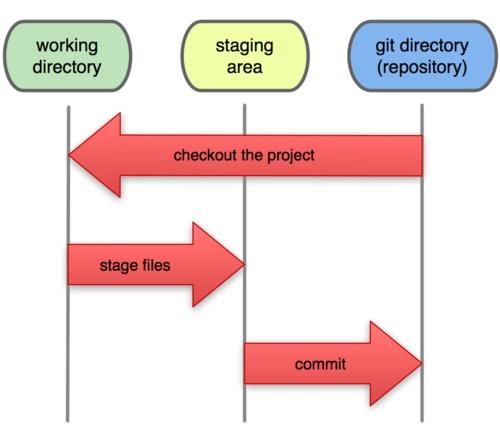
# Experiment 5

**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.

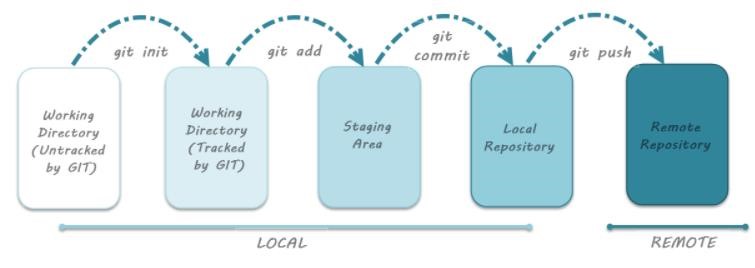
**Staging Area:**

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:** It 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.



# Experiment 6

**Aim:** Add collaborators on GitHub Repo

**Theory:**

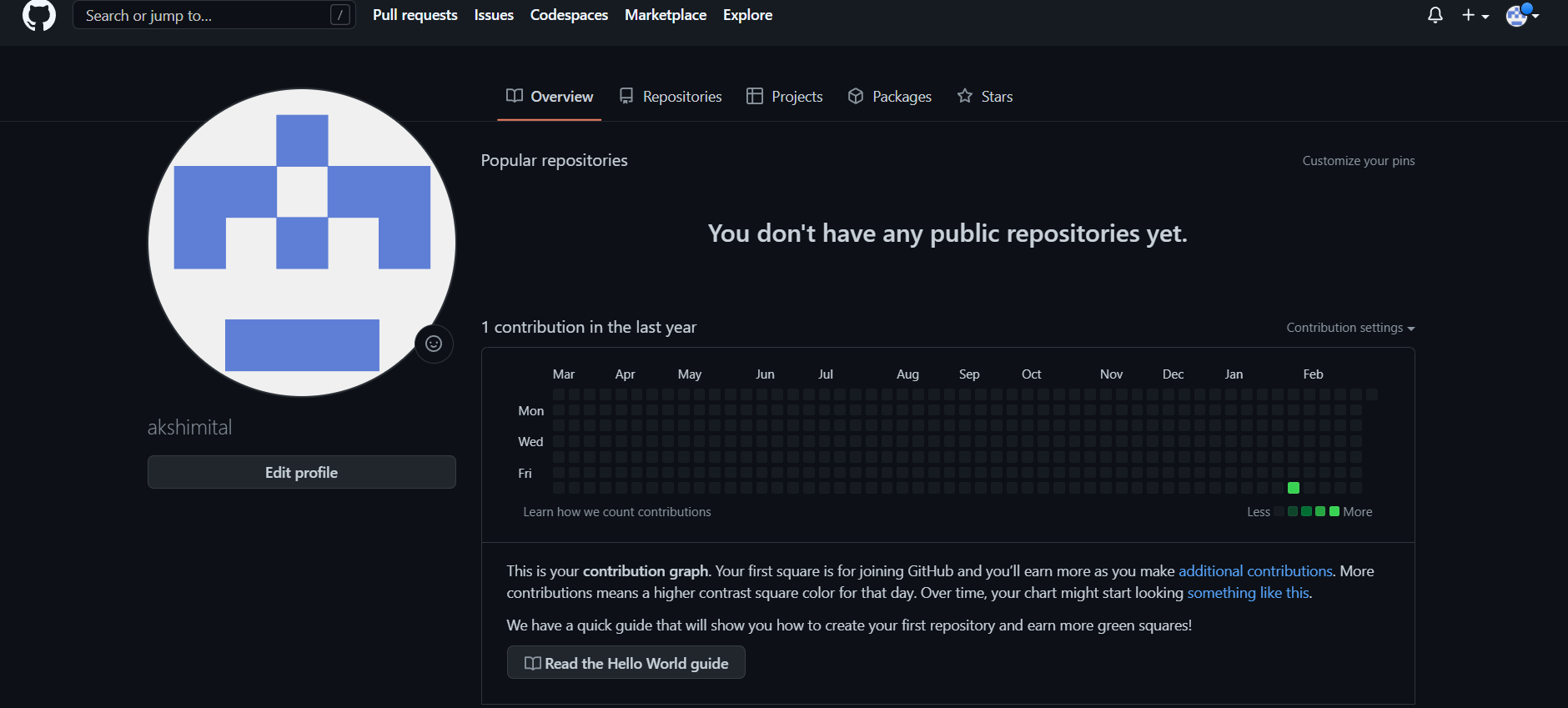
Whenever you make a repository in GitHub, not everyone has the permission to change or push codes into your repository. The users have a read-only access. In order to allow other individuals to make changes to your repository, you need to invite them to collaborate to the project.

GitHub also restricts the number of collaborators we can invite within a period of 24 hours. If we exceed the limit, then either we have to wait for 24-hours or we can also create an organization to collaborate with more people.

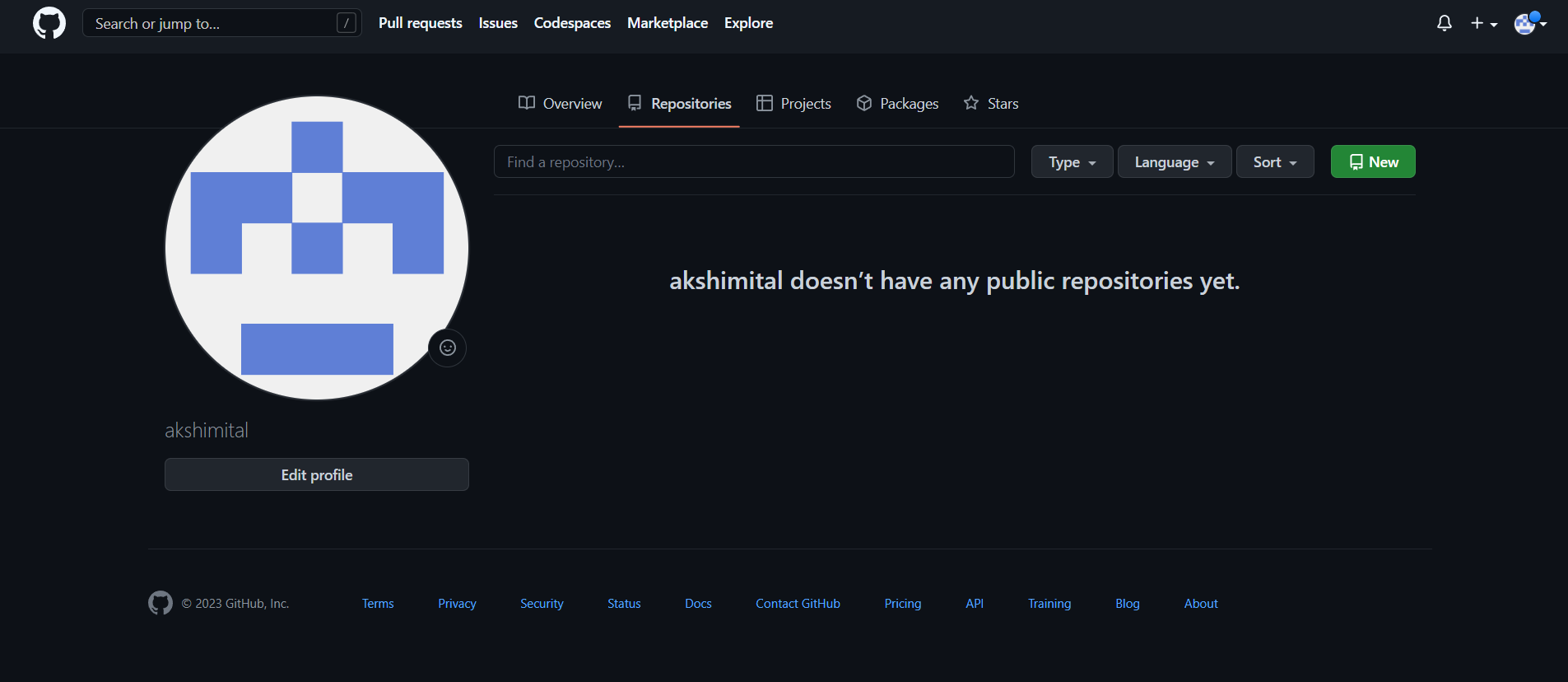
Being a collaborator, the user can create, merge and close pull requests in the repository. They can also remove them as the collaborator.

**Procedure:**

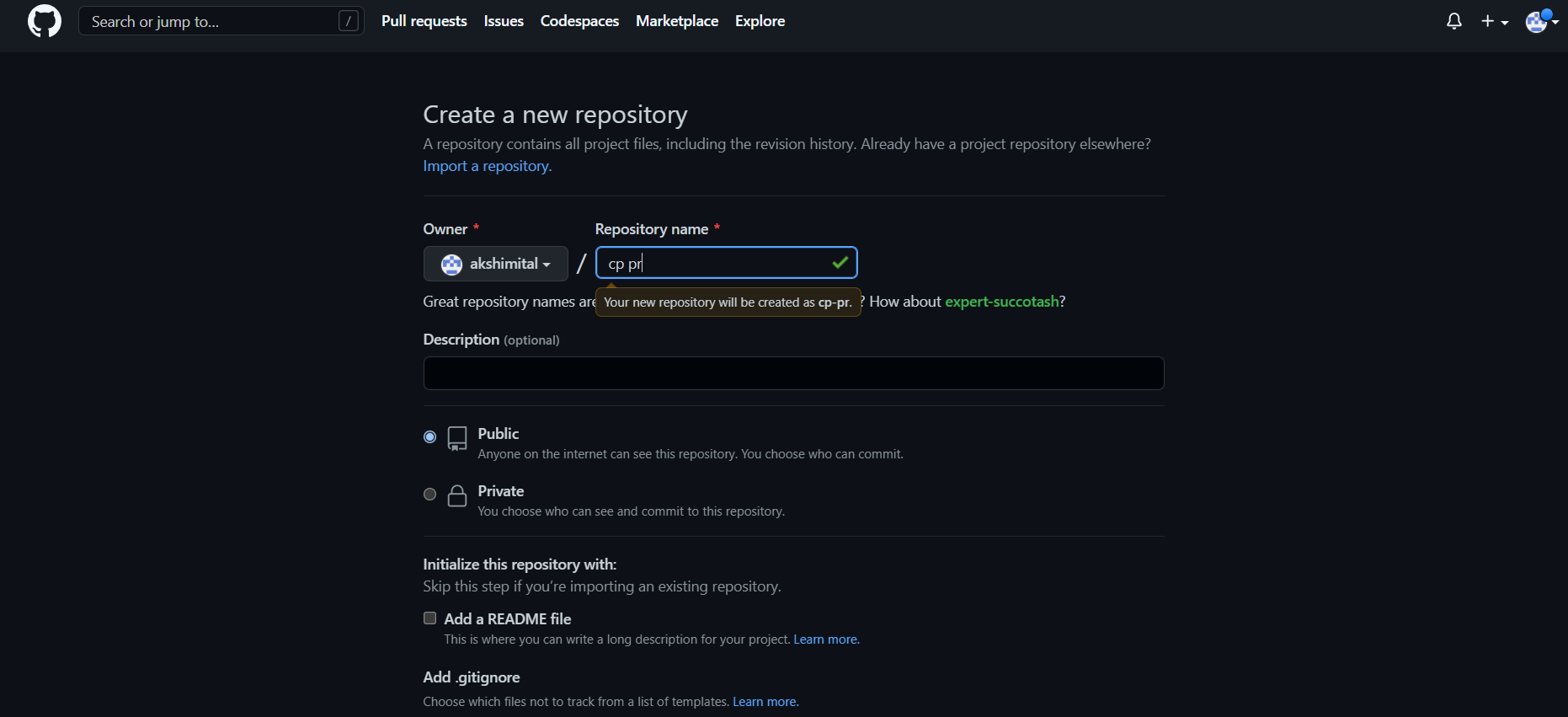
1. Login to your GitHub account and you will land on the homepage as shown below. Click on Repositories option in the menu bar.



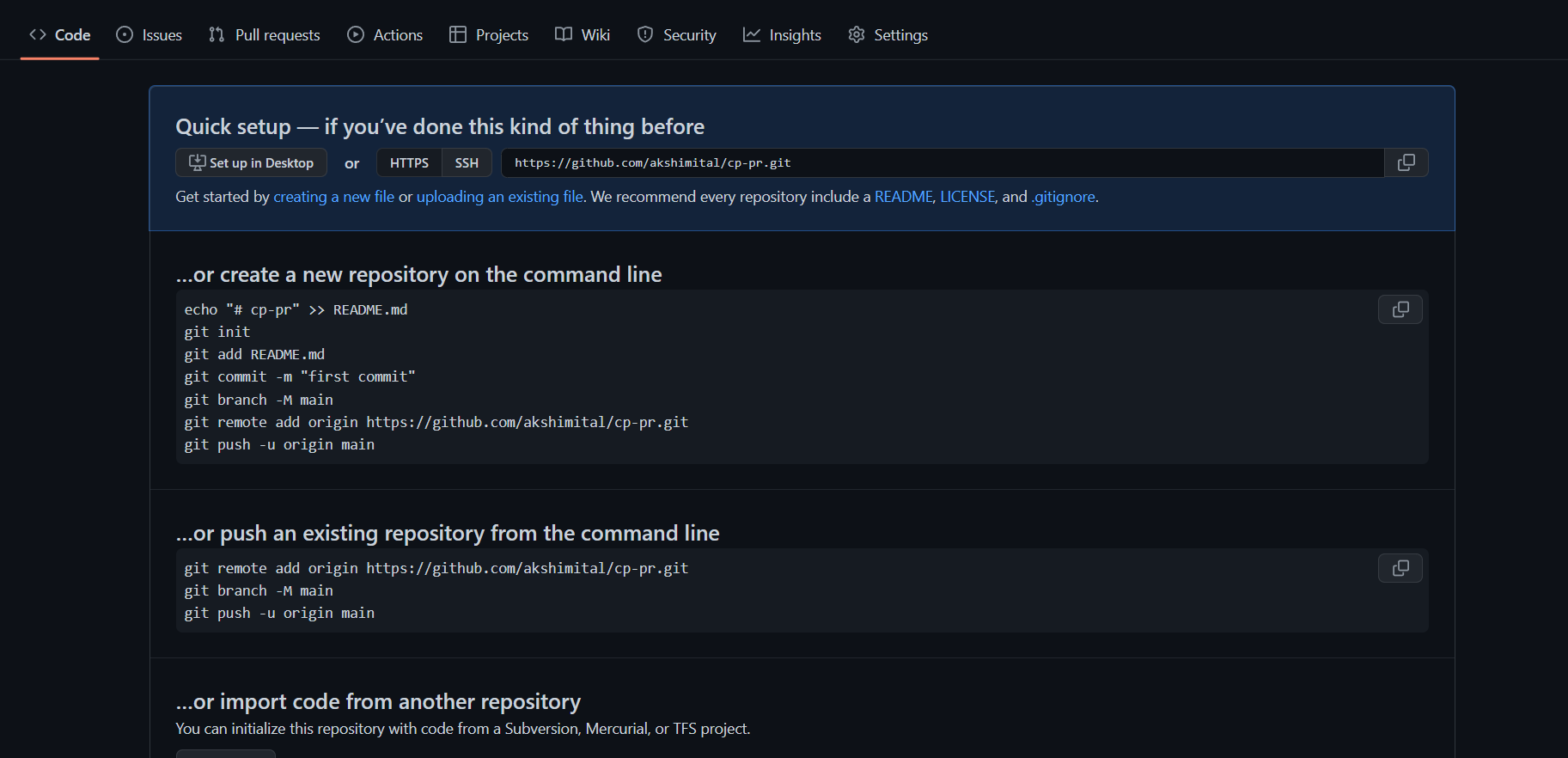
2.Click on the ‘New’ button in the top right corner.



3. Enter the Repository name and add the description of the repository.

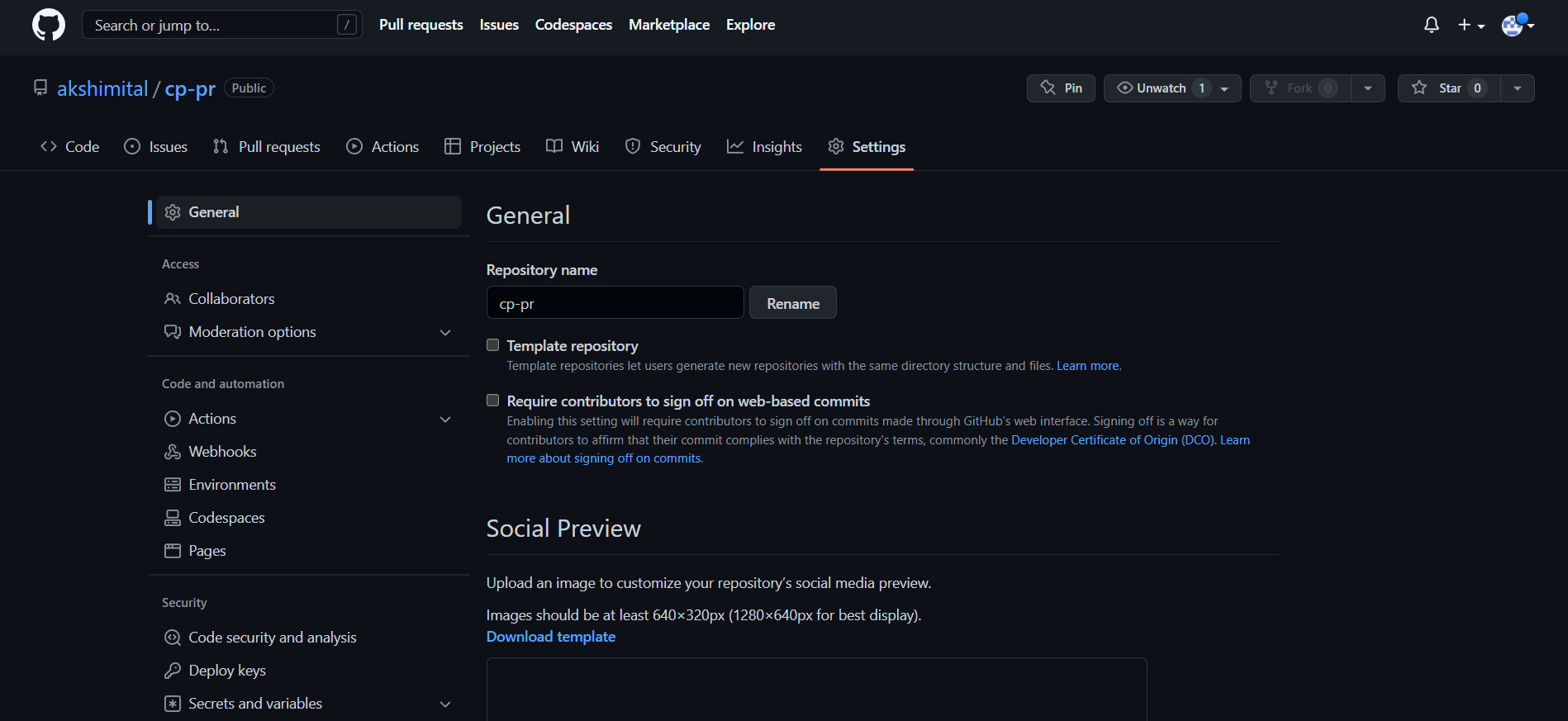
4.Select if you want the repository to be public or private.

5.If you want to import code from an existing repository select the import code option.

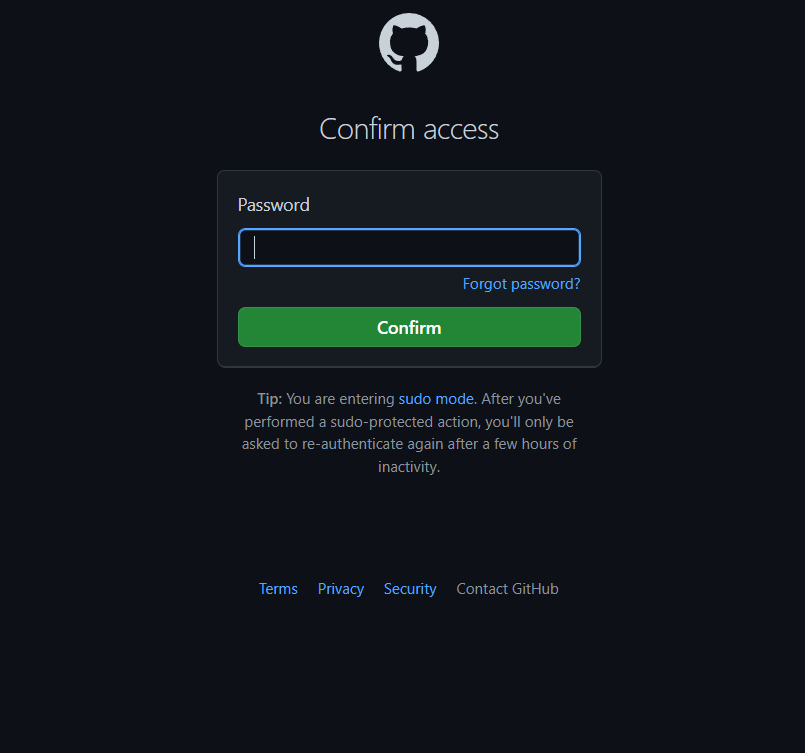


6.Now, you have created your repository successfully.

7.To add collaborators to your repository, open your repository and select settings option in the navigation bar.

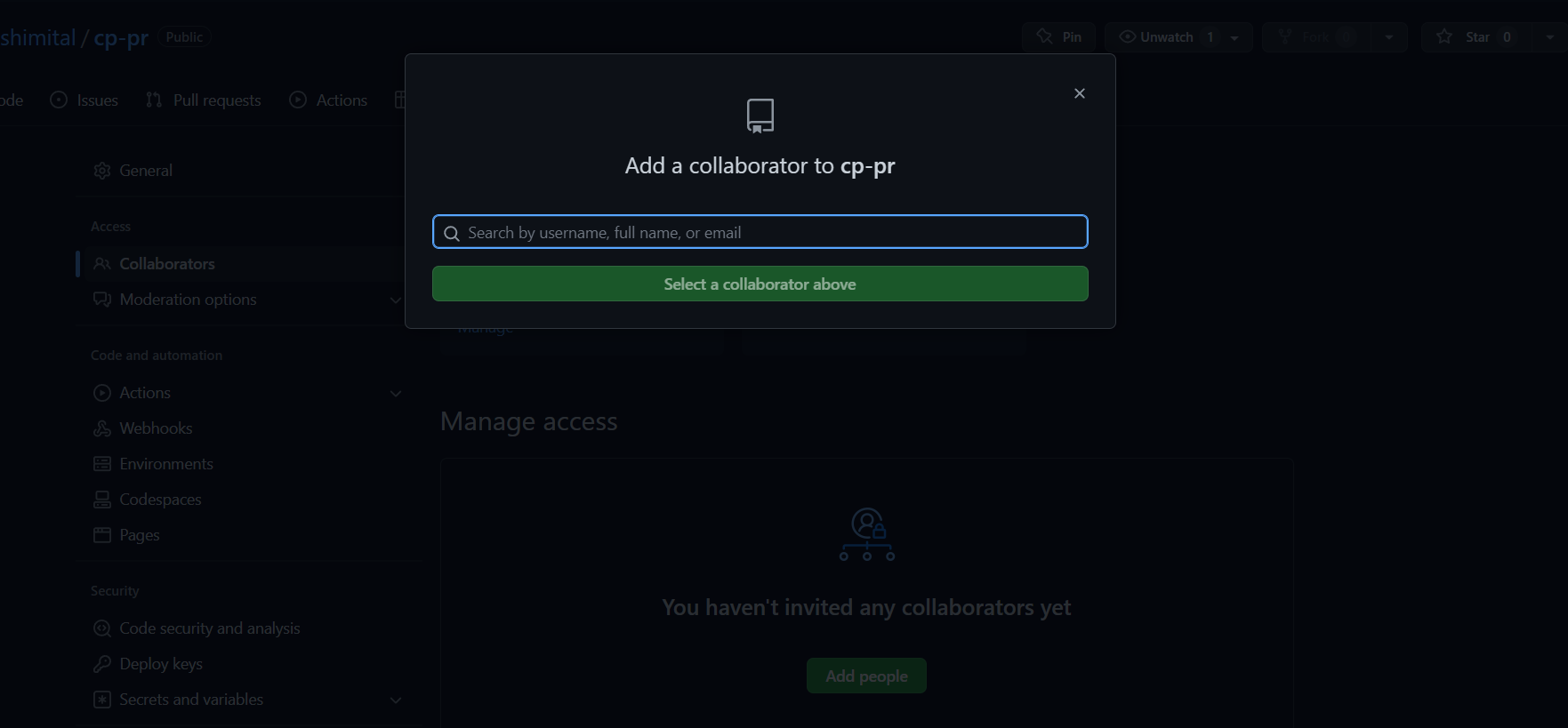


8.Click on Collaborators option under the access tab. 9.After clicking on collaborators, GitHub asks you to enter your password to confirm the access to the repository.



10.After entering the password, you can manage access and add/remove team members to your project.

11.To add members, click on the add people option and search the id of your respective team member.

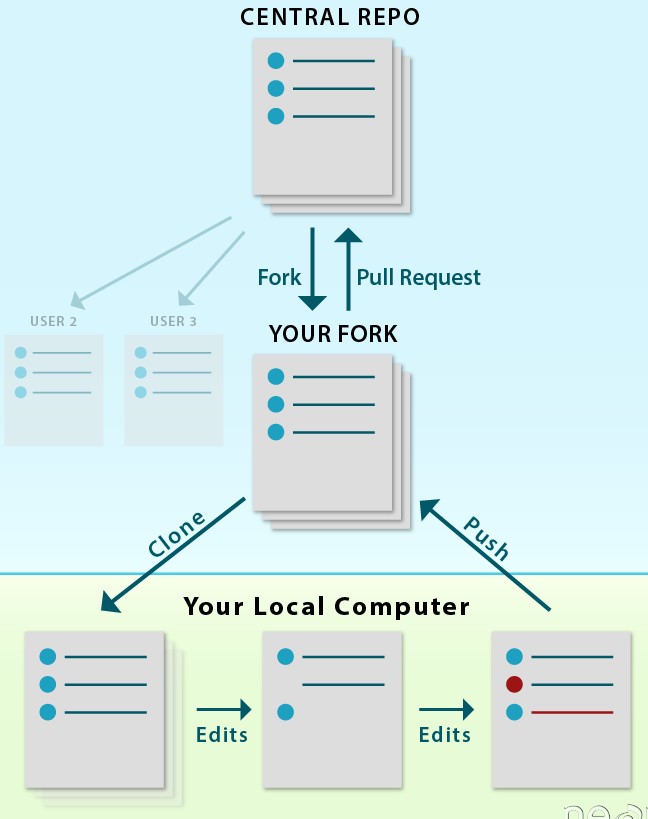


12.To remove any member, click on remove option available in the last column of member’s respective row.

# Experiment 7

## **Aim:** Fork and Commit

**Theory:** A fork is a copy of a repository that you manage. It allows us to freely experiment with the data. After creating a fork, we can make any desired change like adding collaborators, rename files, generate GitHub pages but all these changes won’t be reflected in the original repository.



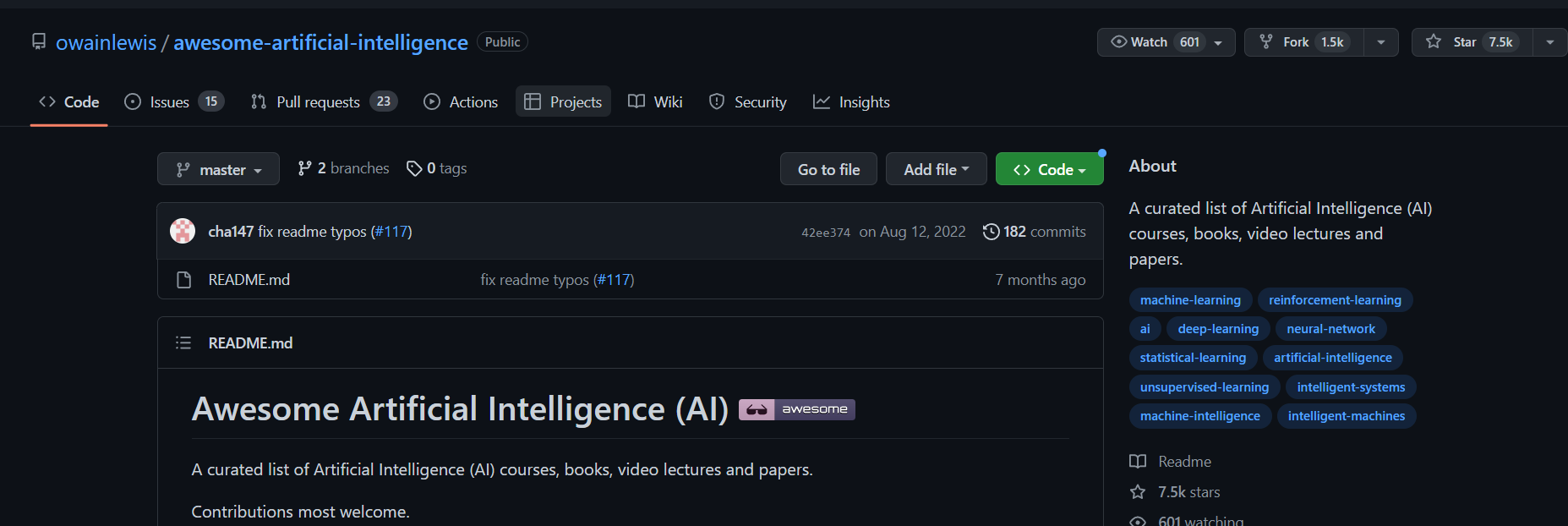
reference for picture: https://www.earthdatascience.org

To import the changes into the original repository, the user needs to send a pull request to the maintainer. If the maintainer closes the pull request only then the content can be added to the original repository.

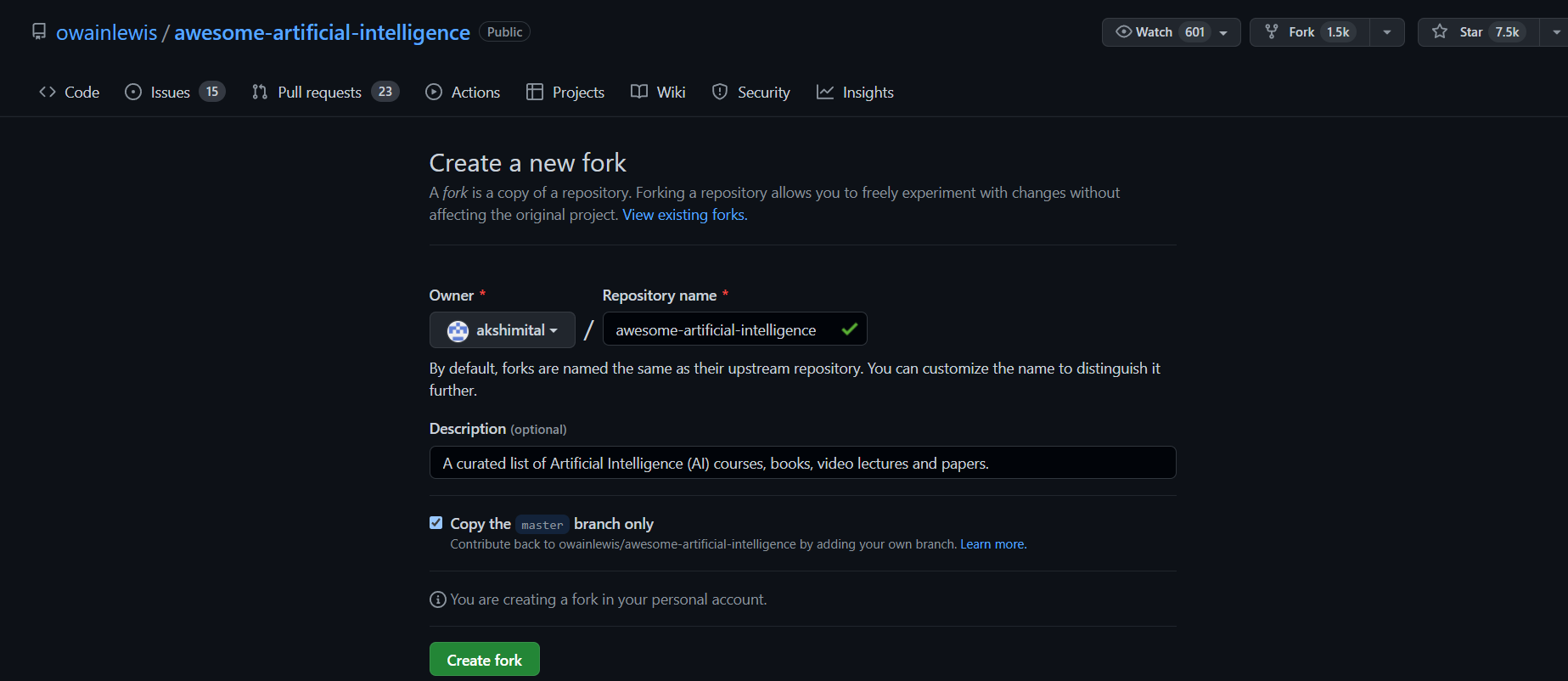
Forking is a better method than directly cloning any repository, as in cloning only the default branch is cloned whereas forking creates a clone of the complete repo and also allows us to push the changes to the main repository by using open and close pull request.

**Procedure:**

1. To fork a repository first thing you need to do is, sign in to your GitHub account and then you come to the repository you want to fork, so here for demo purpose am using **awesome-artificial-intelligence** repository.



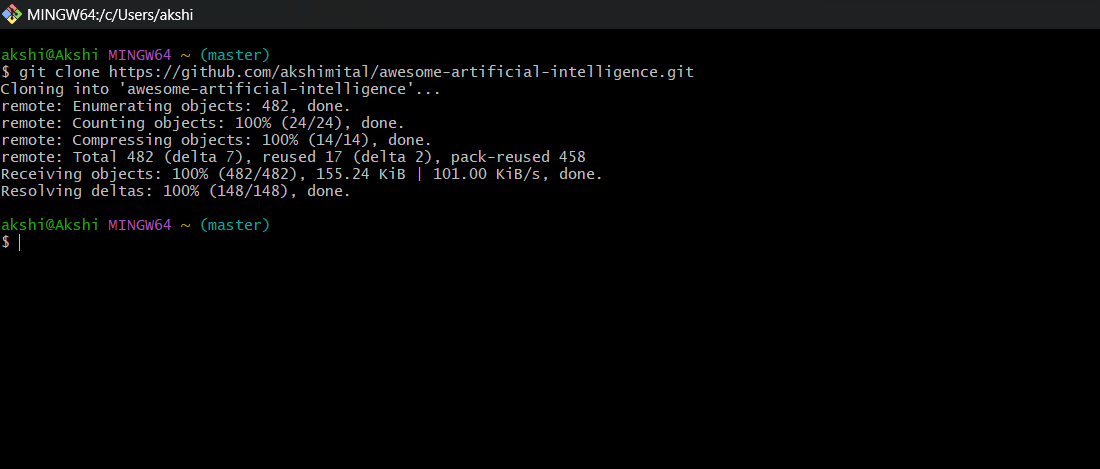
2.Click on the **Fork** button on right upside corner. Then it will ask to create a new fork, add description if you want and then click on create fork.



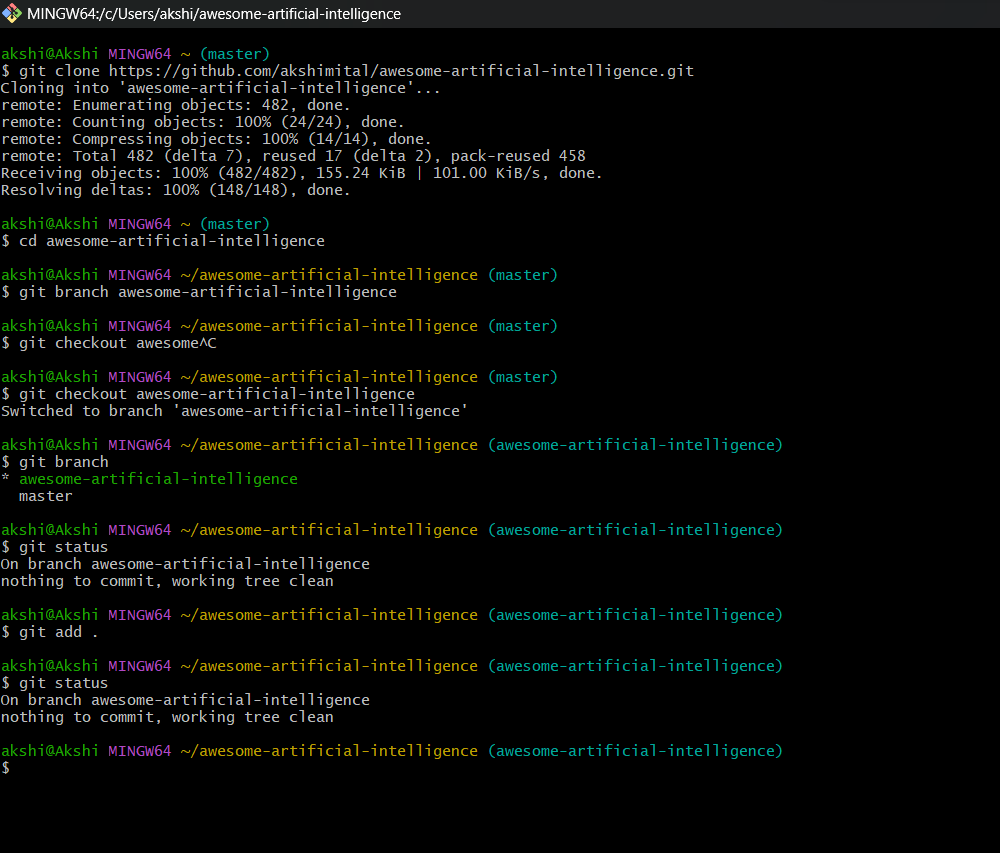
3.Now you will have a copy of the repo you have forked from other user. Now you can do any modification you want without making changes to main source code.

4.Now type git clonehttps://github.com/akshimital/awesome-artificial-intelligence.git on git.

Git clone <url> --> This command is used to fetch the remote repo or to clone the repo.



5.Now Open the file make changes in it and commit it and push it to remote.

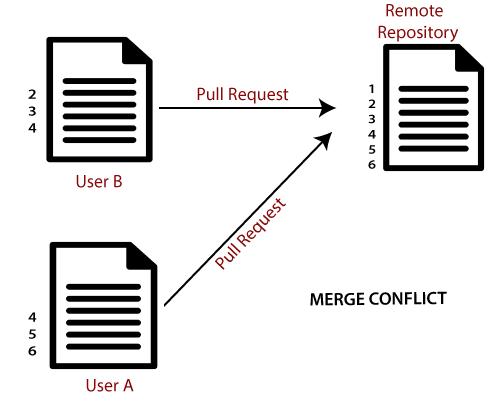


# Experiment 8

**Aim:** Merge and Resolve conflicts created due to own activity and collaborators activity.

**Theory:**

Version control systems are all about managing contributions between multiple distributed authors (usually developers). Sometimes multiple developers may try to edit the same content. If Developer A tries to edit code that Developer B is editing a conflict may occur.



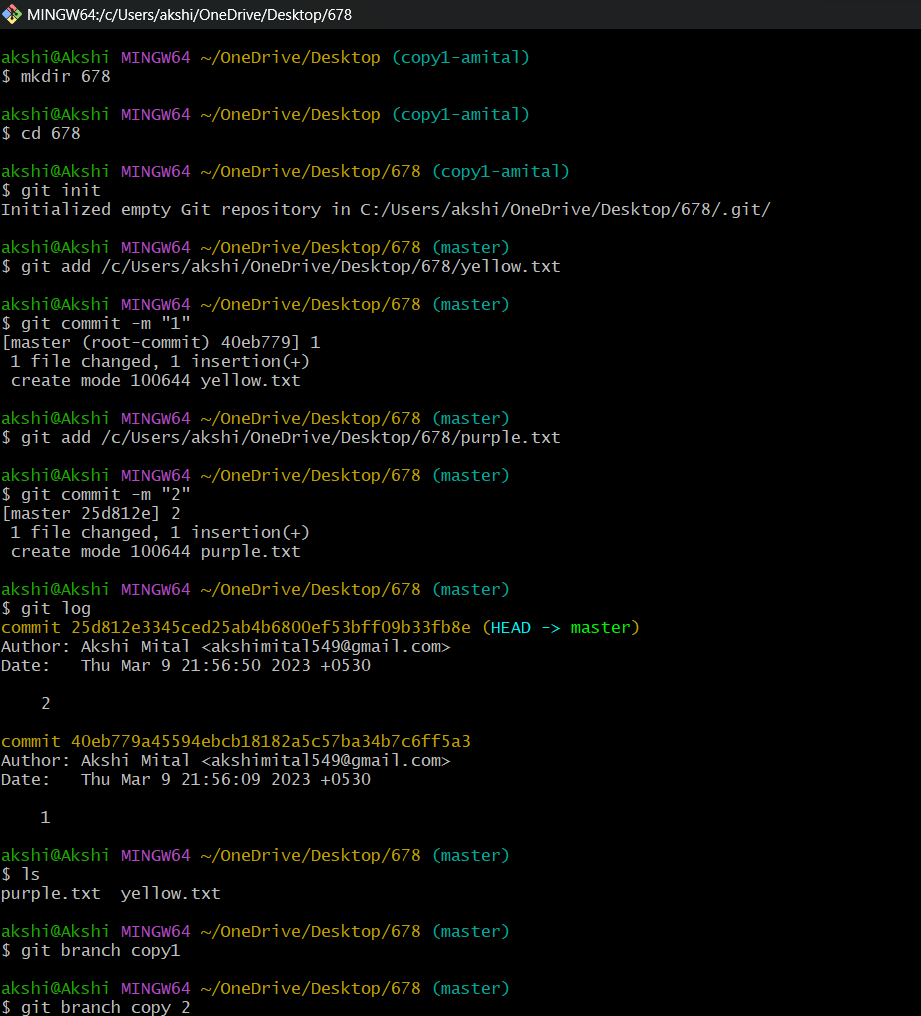
reference for picture: https://www.javatpoint.com/git-merge-and-merge-conflict

If you have a merge conflict on the command line, you cannot push your local changes to GitHub until you resolve the merge conflict locally on your computer.

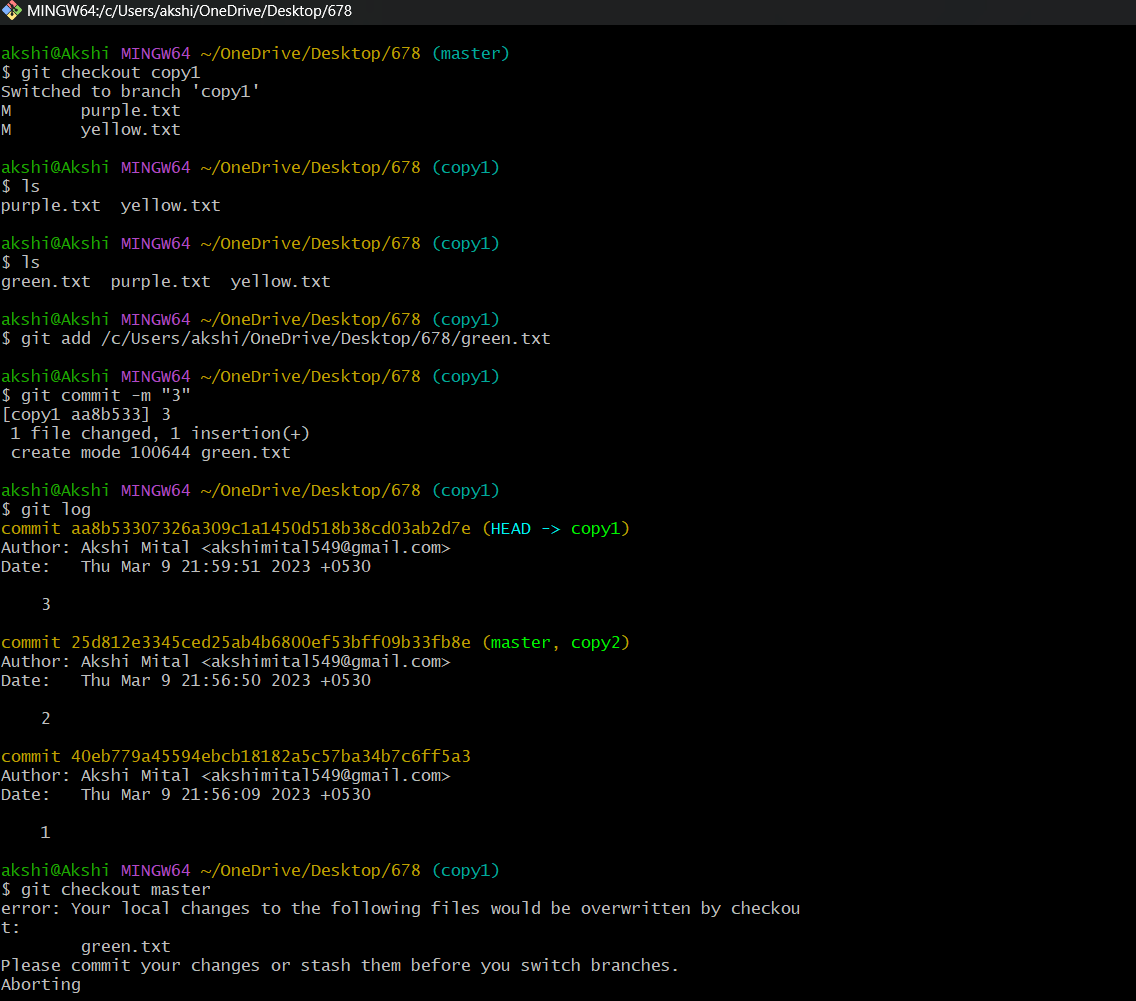
To alleviate the occurrence of conflicts developers will work in separate [isolated branches.](https://www.atlassian.com/git/tutorials/using-branches) If a merge conflict still arises between the compare branch and base branch in your pull request, you can view a list of the files with conflicting changes above the Merge pull request button. The Merge pullrequest button is deactivated until you've resolved all conflicts between the compare branch and base branch.

**Procedure:**

1. Do changes in master branch and commit those change. And checkout to different branch and again do changes and commit it. Now checkout to master branch and merge that branch in master.

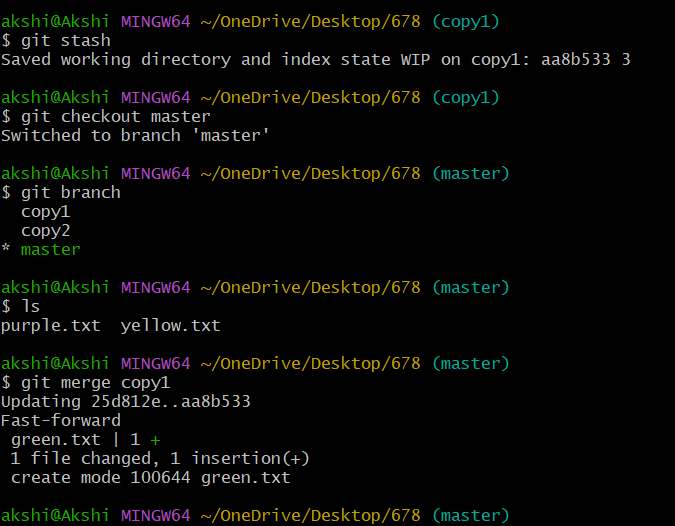


1. Now try to merge it will give Conflicts Error.



1. Use Command “git mergetool” to solve the conflict.

**git -mergetool** – Run merge conflict resolution tools to resolve merge conflicts.



# Experiment 9

**Aim:** Reset and Revert

### Theory:

Git-revert – Revert some existing commits.

A reset is an operation that takes a specified commit and resets the "three trees" to match the state of the repository at that specified commit. A reset can be invoked in three different modes which correspond to the three trees. In reset, rest of the commits wash out after the mentioned commit. This is a limitation of reset command that we cannot have any random access.

A revert is an operation that takes a specified commit and creates a new commit which inverses the specified commit. git revert can only be run at a commit level scope and has no file level functionality.

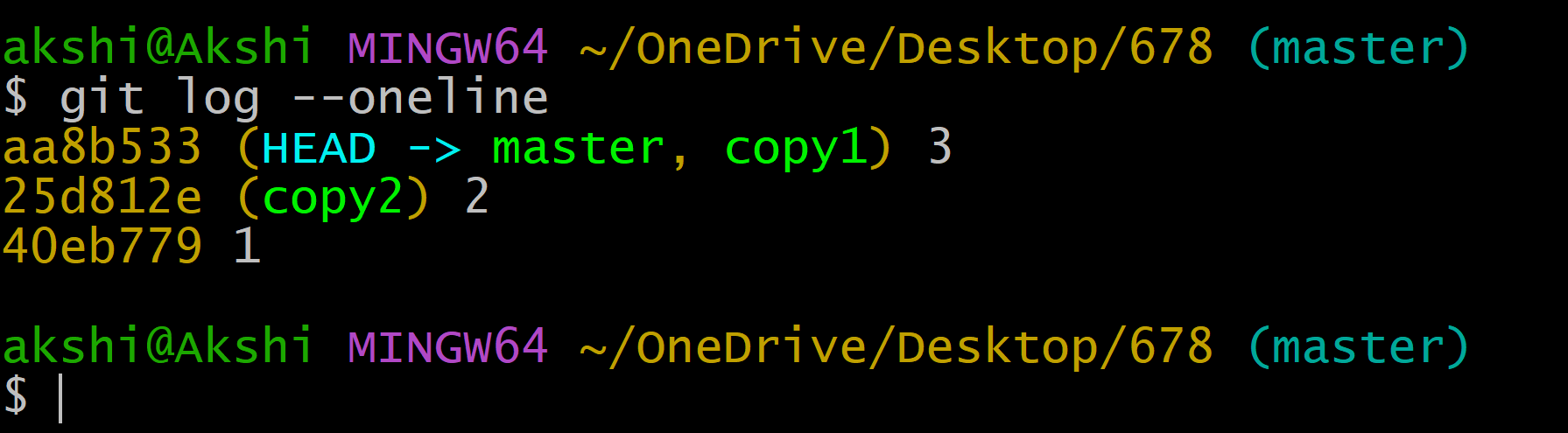
These two features justify the Version- controlled feature of the git as we can rollback to any version at any time.

### PROCEDURE:

Firstly, prepare a log of multiple commits to make the reset and revert command function.

**Reset:** reset is the command we use when we want to move the repository back to a previous commit, discarding any changes made after that commit.

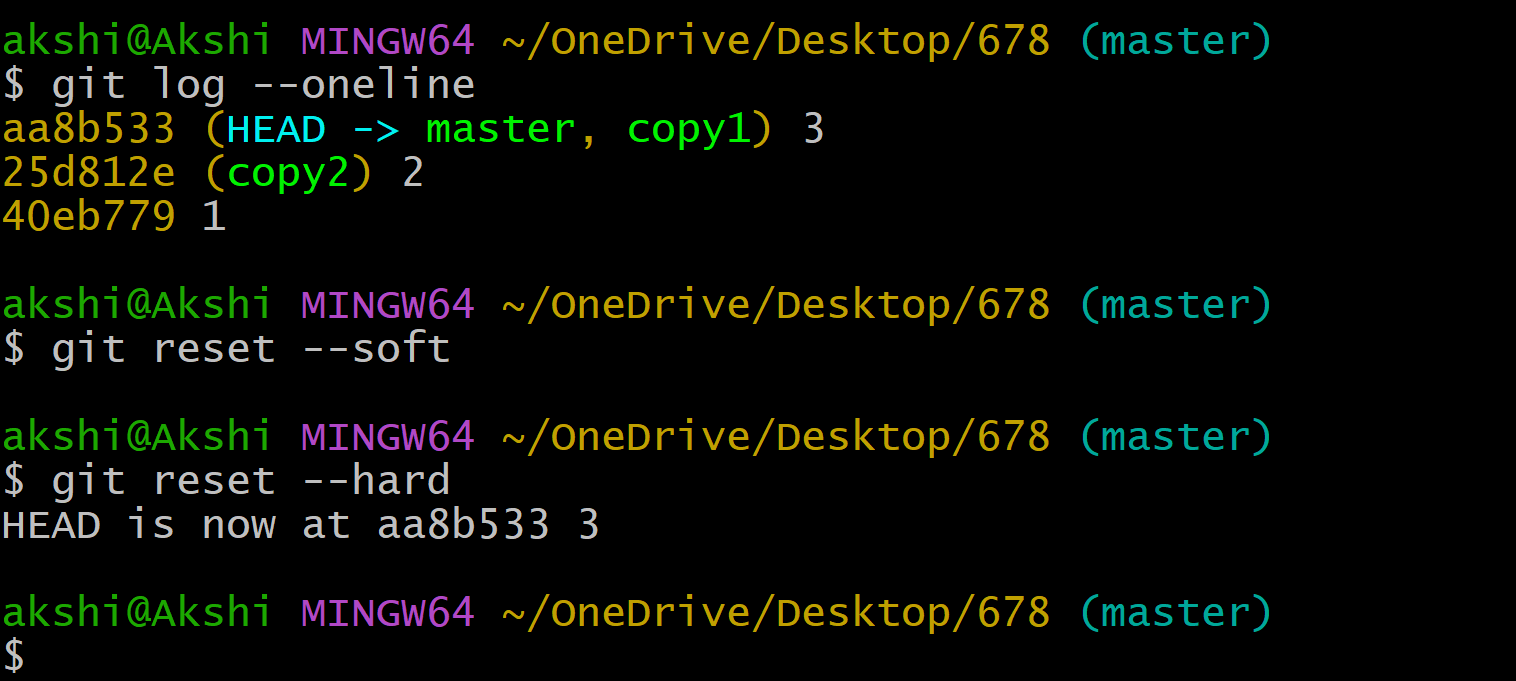
* 1. Create few files, stage them and commit.
  2. Check the git log.



* 1. Pick any commit where you want the repository to rollback. Copy its checksum and paste it in the $ **git reset checksum** command.

The head is now pointing the commit whose checksum we have provided that means the commits that followed vanished.

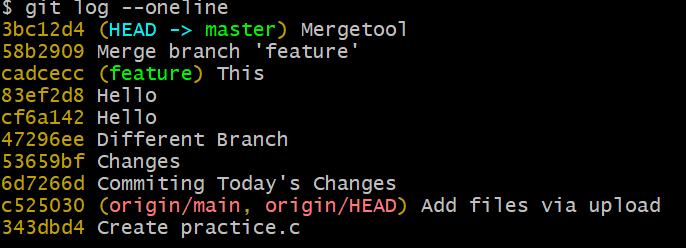
4)In you want undo this change, you copy the checksum of the commit you want back and run the same command again.



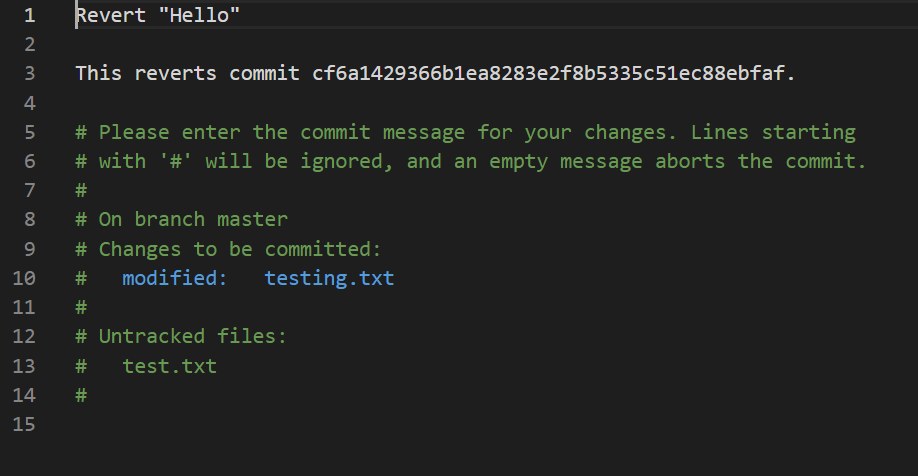
Revert:

Follow these steps to revert any change:

1)Pick the change where you want the project to revert back. Copy its checksum and paste it in the revert command.



2)A window will appear. Press ‘I” and write the statement you want to be displayed for reverting the change.



3)After completing press ‘esc’ and write: wq in the terminal

4)Check the git log and you will find another commit is added without affecting the rest commits.

5)The change associated to the reverted commit has disappeared