

Source Code Management

(CS181)

Submitted by

Team Member 1 . Aaryan Singh 2110990019.

Team Member 2 Abhishek Mittal 2110990067.

Team Member 3 . Abhishek Kumar 2110990065.



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	G-01
Course Coordinator	Dr. Monit Kapoor		

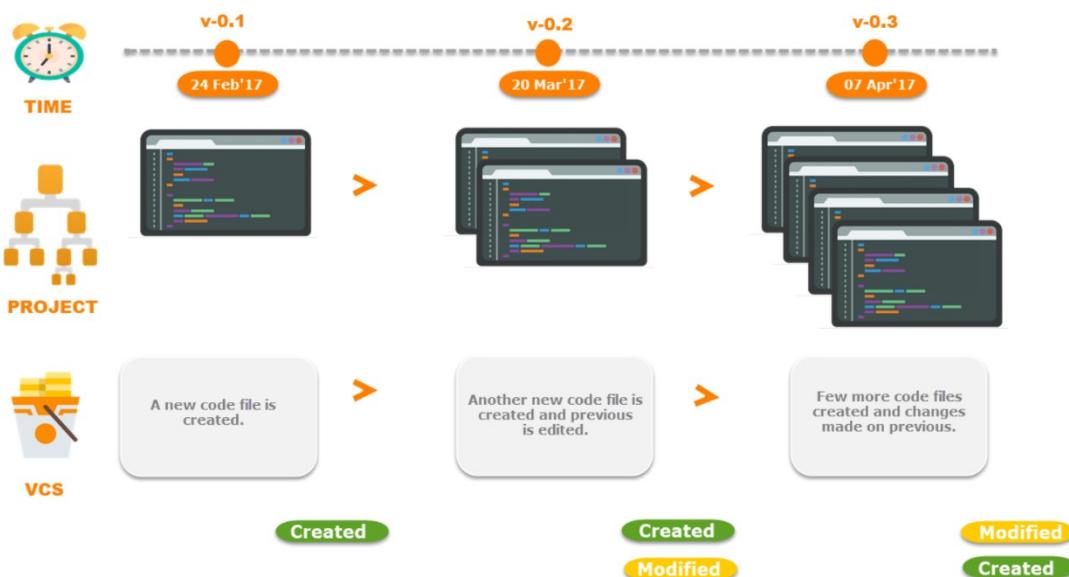
Table of Content

S.No.	Title	Page No.
1	Version control with Git	2-5
2	Problem Statement	6

3	Objective	7
4	Concepts and commands	8-11
5	Workflow and Discussion	12-29
6	Reference	30

1. Version control with Git

What is “version control”, and why should you care? Version control is a system that records changes to a file or set of files over time so that you can recall specific versions later. For the examples in this book, you will use software source code as the files being version controlled, though in reality you can do this with nearly any type of file on a computer. If you are a graphic or web designer and want to keep every version of an image or layout (which you would most certainly want to), a Version Control System (VCS) is a very wise thing to use. It allows you to revert selected files back to a previous state, revert the entire project back to a previous state, compare changes over time, see who last modified something that might be causing a problem, who introduced an issue and when, and more. Using a VCS also generally means that if you screw things up or lose files, you can easily recover. In addition, you get all this for very little overhead.



Local Version Control Systems

Many people’s version-control method of choice is to copy files into another directory (perhaps a time-stamped directory, if they’re clever). This approach is very common because it is so simple, but it is also incredibly error prone. It is easy to forget which directory you’re in and accidentally write to the wrong file or copy over files you don’t mean to. To deal with this

issue, programmers long ago developed local VCSs that ~~CHAKRA~~ simple database that kept all the changes to files under revision control.

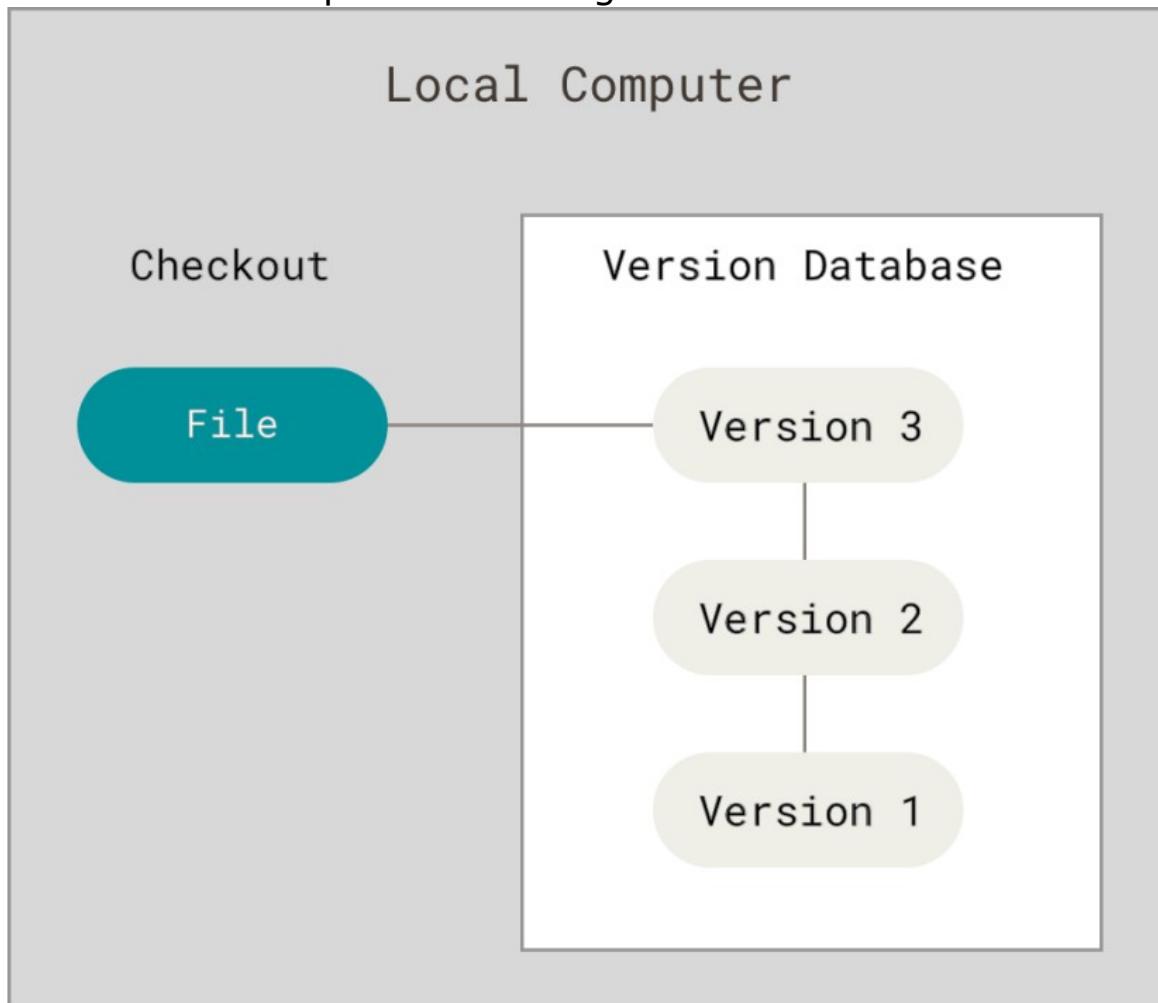


Figure 1. Local version control

One of the most popular VCS tools was a system called RCS, which is still distributed with many computers today. RCS works by keeping patch sets (that is, the differences between files) in a special format on disk; it can then re-create what any file looked like at any point in time by adding up all the patches.

Centralized Version Control Systems

The next major issue that people encounter is that they need to collaborate with developers on other systems. To deal with this problem, Centralized Version Control Systems (CVCSs) were developed. These systems (such as

CVS, Subversion, and Perforce) have a single server that ~~the UNIVERSITY~~ ~~install~~ all the versioned files, and a number of clients that check out files from that central place. For many years, this has been the standard for version control.

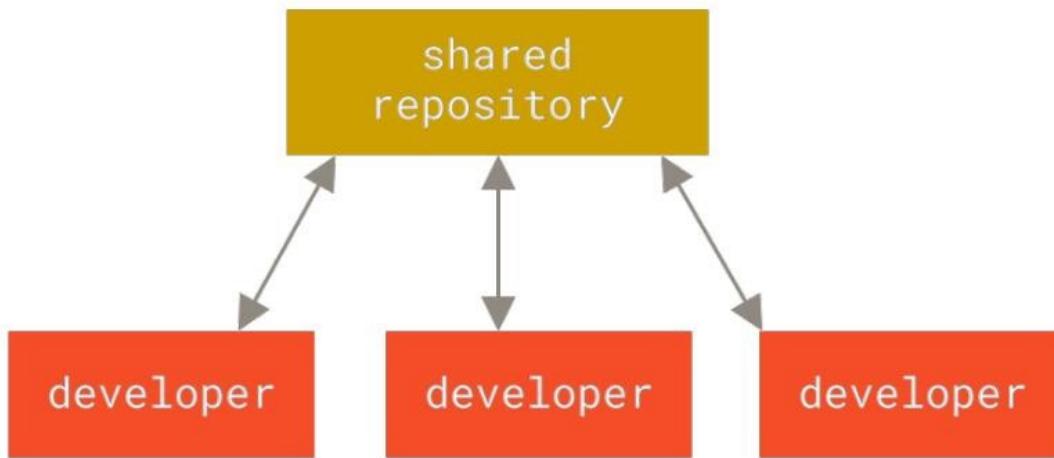


Figure 2. Centralized version control

This setup offers many advantages, especially over local VCSs. For example, everyone knows to a certain degree what everyone else on the project is doing. Administrators have fine-grained control over who can do what, and it's far easier to administer a CVCS than it is to deal with local databases on every client. However, this setup also has some serious downsides. The most obvious is the single point of failure that the centralized server represents. If that server goes down for an hour, then during that hour nobody can collaborate at all or save versioned changes to anything they're working on. If the hard disk the central database is on becomes corrupted, and proper backups haven't been kept, you lose absolutely everything — the entire history of the project except whatever single snapshots people happen to have on their local machines. Local VCSs suffer from this same problem — whenever you have the entire history of the project in a single place, you risk losing everything.

Distributed Version Control Systems

This is where Distributed Version Control Systems (DVCSs) step in. In a DVCS (such as Git, Mercurial, Bazaar or Darcs), clients don't just check out the latest snapshot of the files; rather, they fully mirror the repository, including its full history. Thus, if any server dies, and these systems were collaborating via that server, any of the client repositories can be copied

back up to the server to restore it. Every clone is really a full backup of all the data.

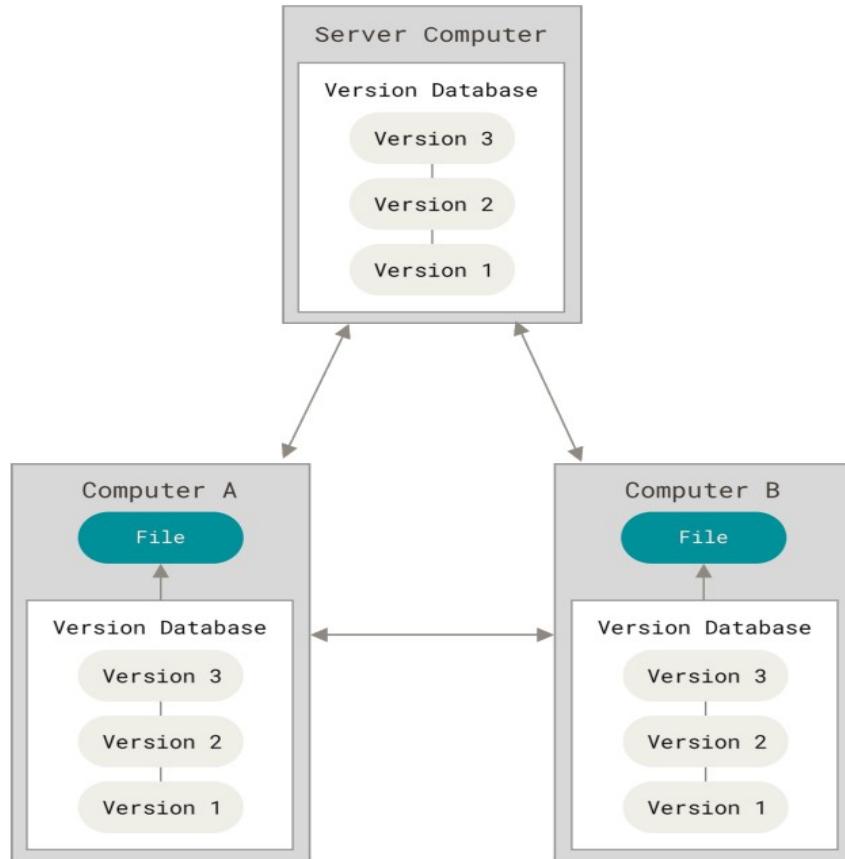


Figure 3. Distributed version control

Furthermore, many of these systems deal pretty well with having several remote repositories they can work with, so you can collaborate with different groups of people in different ways simultaneously within the same project. This allows you to set up several types of workflows that aren't possible in centralized systems, such as hierarchical models.

Why Do We Need It?

Complex code development is impossible without a Version Control System (VCS). Version Control is used to track and control changes to source code. It is an essential tool to ensure the integrity of the codebase.

Why is Version Control Essential?

- Easy Modification of Codebase
- Reverting Error
- Improves visibility.
- Accelerates product delivery.

3

Version control system keeps track on changes made on a particular software and take a snapshot of every modification .Let's suppose if a team of developer add some new functionalities in a application and the updated version is not working properly so as version control system keeps track of our work so with the help of version control system we can omit the new changes and continue with the previous version .

Benefits of the version control system:

- Enhances the project development speed by providing efficient collaboration,
- Leverages the productivity, expedites product delivery, and skills of the employees through better communication and assistance,
- Reduce possibilities of errors and conflicts meanwhile project development through traceability to every small change,
- Employees or contributors of the project can contribute from anywhere irrespective of the different geographical locations through this VCS,
- For each different contributor to the project, a different working copy is maintained and not merged to the main file unless the working copy is validated. The most popular example is Git, Helix core, Microsoft TFS,
- Helps in recovery in case of any disaster or contingent situation,
- Informs us about Who, What, When, Why changes have been made.

4. Concepts and Commands

❖ Here are some commands which is used in GIT.

git config --global user.name “name”:

This command sets the author’s name to be used with your commits.

git config --global user.email “email”: This command sets the author’s email address to be used with your commits.

git --version :

It is used to display the version of git bash.

pwd:

Gives the full pathname of the current working directory to the standard output

git init :

Used to initialize the repository.

touch “filename”: It creates a new file.

Vi “filename”:

It creates and add content to a file.

git add -a:

Moves changes from the working directory to the staging area.

git commit -m “info of commit”:

This command records or snapshots the file permanently in the version history.

ls :

It gives the name of file present in the current folder.

git status:

This command lists all the files that have to be committed.

rm -rf .git:

This command is used delete the .git hidden folder.

git rm [file] :

This command deletes the file from your working directory and stages the deletion.

git log :

This command is used to list the version history for the current branch.

git branch:

This command lists all the local branches in the current repository.

git branch [branch name] :

This command creates a new branch.

git checkout [branch name] :

This command is used to switch from one branch to another.

git branch -m [branch name] :

This command is used to rename the branch.

git merge [branch name] :

This command is used to merge a command.

git branch -d [branch name] :

This command is used to soft delete a branch.

git branch -D [branch name] :

When you want to delete a branch without merging it. So, we use this command.

git log --oneline :

It gives the checkout information in one line.

git mv [file name] [file name] :

This command is used to rename a file.

git restore --staged [file name] :

This command is used to remove from staging area.

git log --oneline --graph :

It gives the checkout in the form of network graph.

git clone [URL] :

git remote:

This command is used to connect your local repository to the remote server.

git push :

This command is used to push our repository.

git pull :

This command is used to pull a repository.

git remote remove [remote name] :

This command is to remove the remote.

git reset [checksome] :

This command is used to restore the commit upto the checksome.

git revert [checksome] :

This command is used to restore the commit upto the checksome and added a new checksome of revert.

.gitignore.txt :

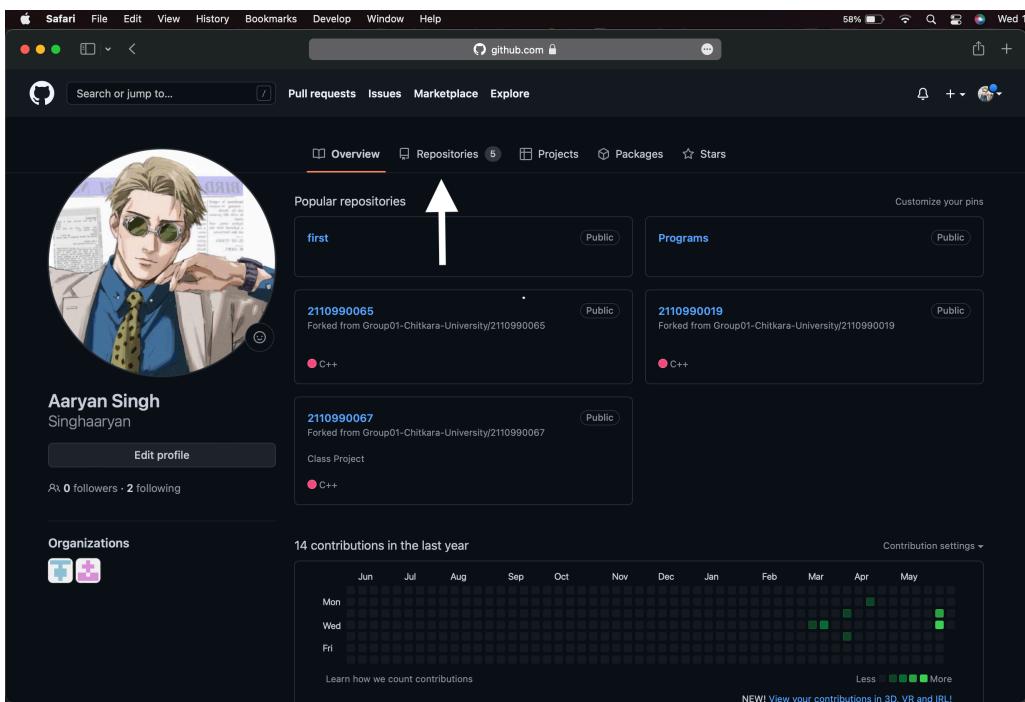
It is a text file used to ignore files which is not usable for us.

It ignore all the files which is inside .gitignore file.

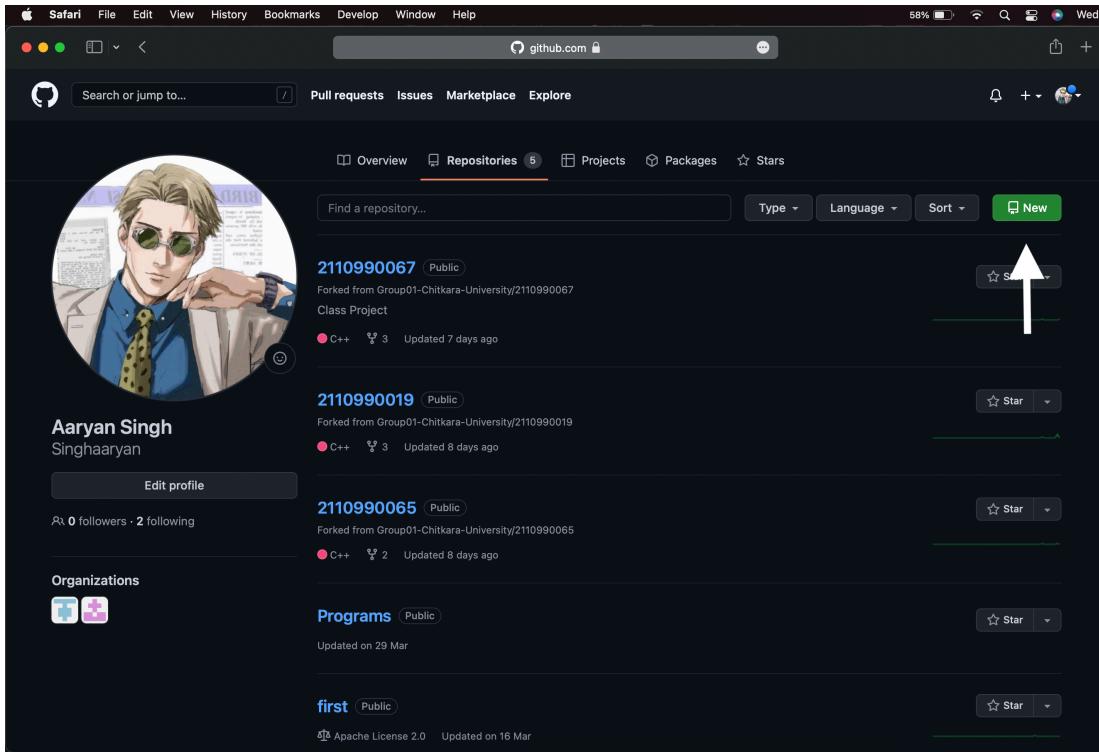
5.Workflow and Discussion.

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

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



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

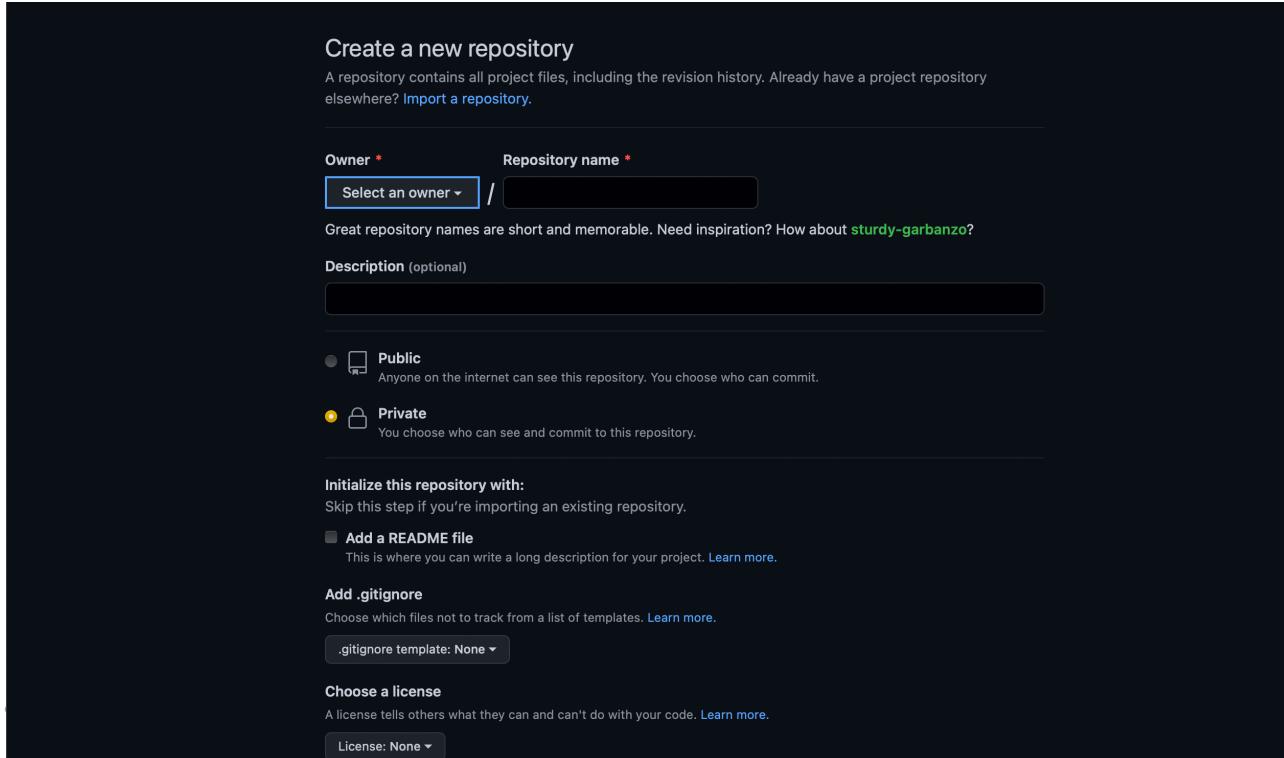


A screenshot of a GitHub profile page for 'Aaryan Singh'. The profile picture is a stylized drawing of a man with blonde hair and sunglasses. The page shows several repositories under the 'Repositories' tab:

- 2110990067** (Public) - Forked from Group01-Chitkara-University/2110990067. Class Project. C++ 3. Updated 7 days ago.
- 2110990019** (Public) - Forked from Group01-Chitkara-University/2110990019. C++ 3. Updated 8 days ago.
- 2110990065** (Public) - Forked from Group01-Chitkara-University/2110990065. C++ 2. Updated 8 days ago.
- Programs** (Public) - Updated on 29 Mar.
- first** (Public) - Apache License 2.0. Updated on 16 Mar.

The GitHub interface includes a search bar, navigation tabs (Overview, Repositories, Projects, Packages, Stars), and dropdown menus for Type, Language, and Sort. A green 'New' button is located at the top right of the repository list.

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



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository](#).

Owner * **Repository name ***

Select an owner /

Great repository names are short and memorable. Need inspiration? How about [sturdy-garbanzo](#)?

Description (optional)

Public
Anyone on the internet can see this repository. You choose who can commit.

Private
You choose who can see and commit to this repository.

Initialize this repository with:
Skip this step if you're importing an existing repository.

Add a README file
This is where you can write a long description for your project. [Learn more](#).

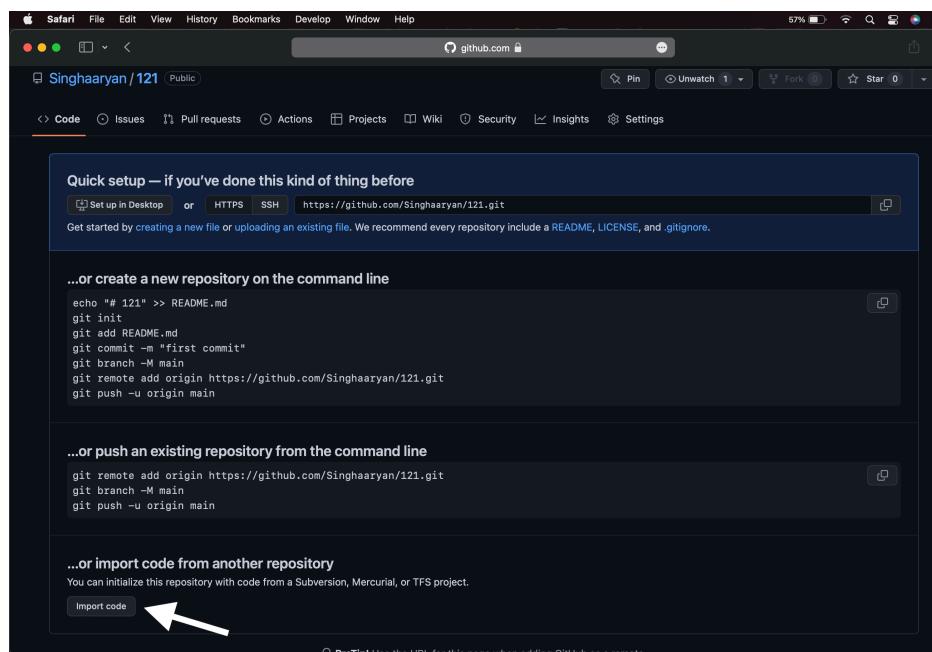
Add .gitignore
Choose which files not to track from a list of templates. [Learn more](#).

.gitignore template:

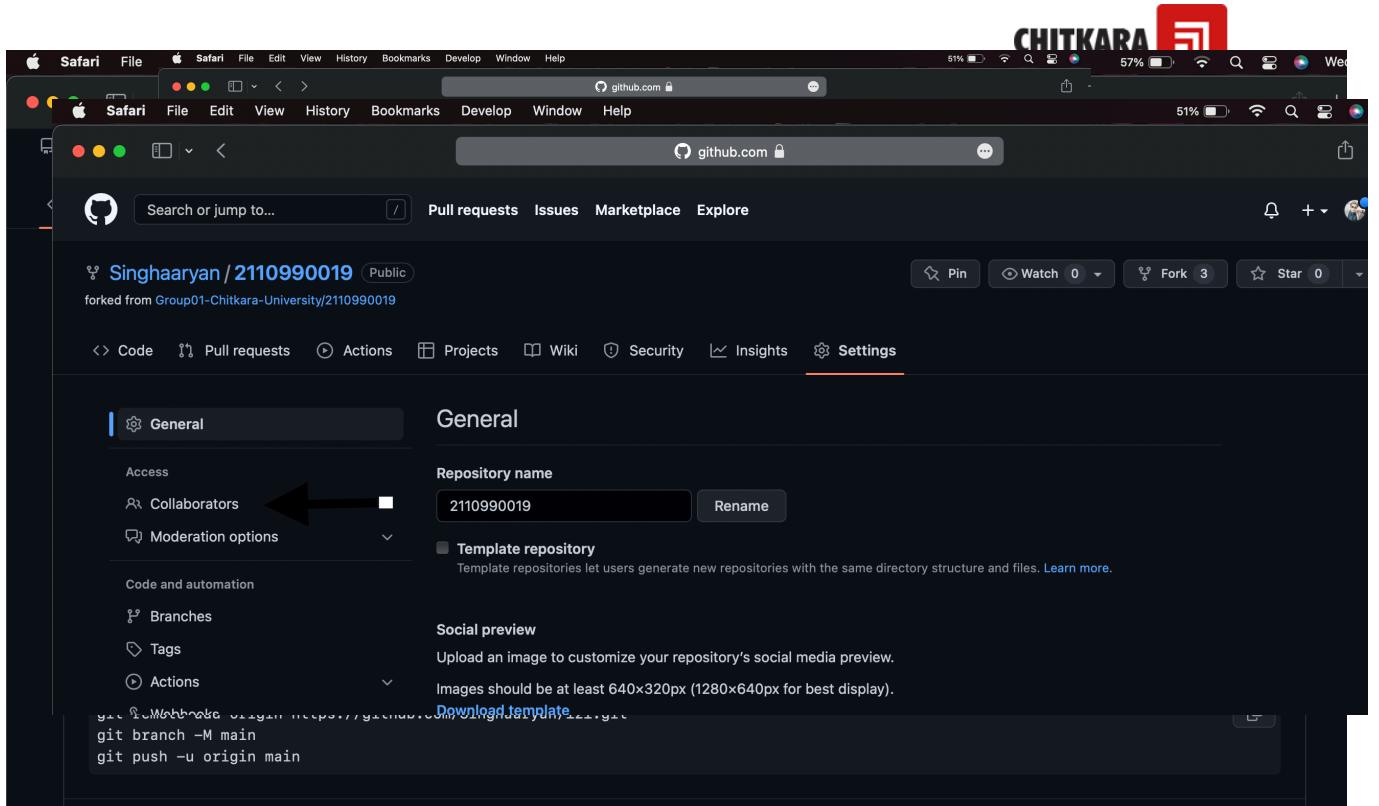
Choose a license
A license tells others what they can and can't do with your code. [Learn more](#).

License:

- Select if you want the repository to be public or private.
- If you want to import code from an existing repository select the import code option.



- To create a new file or upload an existing file into your repository select the option in the following box.
- Now, you have created your repository successfully.
- To add members to your repository, open your repository and select settings option in the navigation bar.



- Click on Collaborators option under the access tab.
- After clicking on collaborators GitHub asks you to enter your password to confirm the access to the repository.
- After entering the password you can manage access and add/remove team members to your project.
- To add members click on the add people option and search the id of your respective team member.

The screenshot shows the GitHub repository settings page for 'Singhaaryan / 2110990019'. The left sidebar includes sections for General, Access (selected), Collaborators (highlighted), Moderation options, Code and automation (Branches, Tags, Actions, Webhooks, Environments, Pages), Security (Code security and analysis, Deploy keys, Secrets), and Integrations (GitHub apps). The main content area displays 'Who has access' with sections for PUBLIC REPOSITORY (repository is public and visible to anyone) and DIRECT ACCESS (0 collaborators have access to this repository, only the owner can contribute). Below this is the 'Manage access' section, which shows a placeholder message: 'You haven't invited any collaborators yet' with a 'Add people' button.

- To remove any member click on remove option available in the last column of member's respective row.
- To accept the invitation from your team member, open your email registered with GitHub.
- You will receive an invitation mail from the repository owner. Open the email and click on accept invitation.

A screenshot of a GitHub repository settings page. The repository is named "Singhaaryan / 2110990019" and is public. The "Settings" tab is selected. On the left, there's a sidebar with sections like General, Access (Collaborators selected), Code and automation, Security, Integrations, and GitHub apps. The main area shows "Who has access" with two sections: "PUBLIC REPOSITORY" (This repository is public and visible to anyone.) and "DIRECT ACCESS" (1 has access to this repository. 0 collaborators. 1 invitation.). Below this is a "Manage access" section where "Abhishek Mittal" is listed with a pending invite. A green "Add people" button is at the top right of this section.

- You will be redirected to GitHub where you can either select to accept or decline the invitation.

A screenshot of a GitHub invitation confirmation page. The URL is https://github.com/Singhaaryan/2110990019/invitations. It shows a message from "Singhaaryan" inviting "Abhishek Mittal" to collaborate. There are "Accept invitation" and "Decline" buttons. Below the buttons, a note says "Owners of 2110990019 will be able to see:" followed by a list of permissions: "Your public profile information", "Certain activity within this repository", "Country of request origin", "Your access level for this repository", and "Your IP address". At the bottom, there's a link to report spam and a "Block Singhaaryan" button.

- You will be shown the option that you are now allowed to push

You now have push access to the Singhaaryan/2110990019 repository.

- Now all members are ready to contribute

Aim: Open and close a Pull Request

- To open a pull request we first have to make a new branch, by using git branch branchname option.

```
Last login: Tue May 24 21:59:17 on ttys003
[senpai@Aaryans-MacBook-Air ~ % pwd
/Users/senpai
[senpai@Aaryans-MacBook-Air ~ % cd desktop
[senpai@Aaryans-MacBook-Air desktop % git clone https://github.com/Singhaaryan/2110990065.git
Cloning into '2110990065'...
remote: Enumerating objects: 10, done.
remote: Counting objects: 100% (10/10), done.
remote: Compressing objects: 100% (9/9), done.
remote: Total 10 (delta 0), reused 10 (delta 0), pack-reused 0
Receiving objects: 100% (10/10), 570.17 KiB | 1.54 MiB/s, done.
[senpai@Aaryans-MacBook-Air desktop % git branch
* main
[senpai@Aaryans-MacBook-Air desktop % pwd
/Users/senpai/desktop
[senpai@Aaryans-MacBook-Air desktop % cd 2110990065
[senpai@Aaryans-MacBook-Air 2110990065 % git branch
* master
[senpai@Aaryans-MacBook-Air 2110990065 % git branch aaryan
senpai@Aaryans-MacBook-Air 2110990065 %
```

- 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 new branch to the main repository.

```

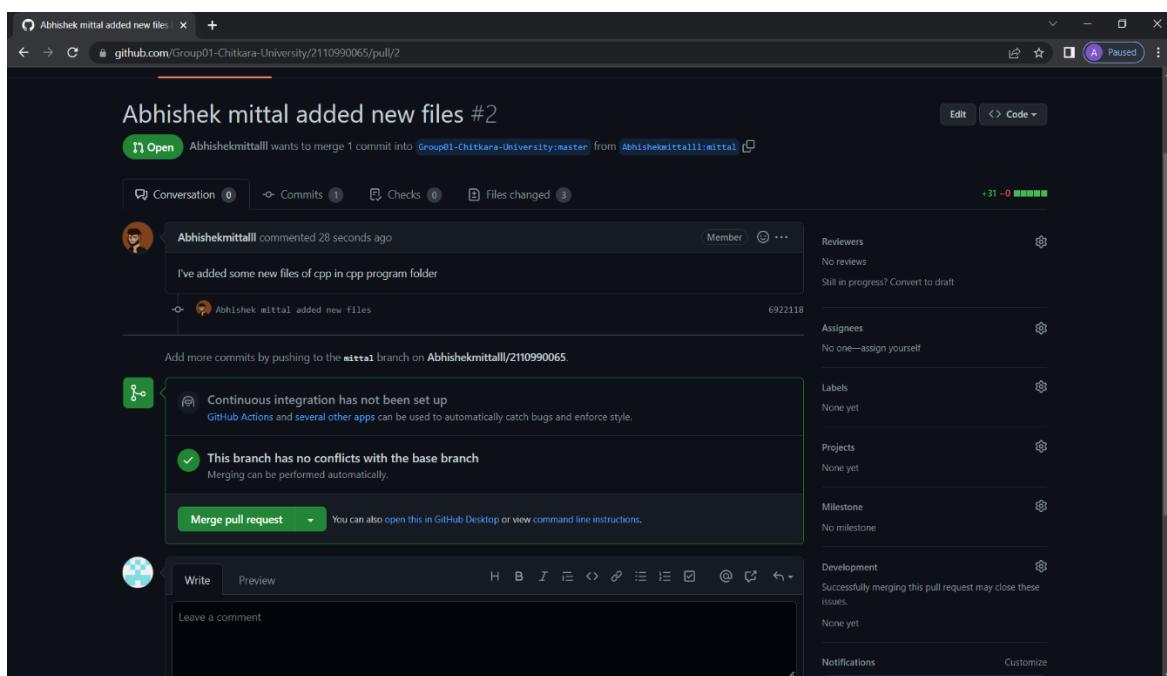
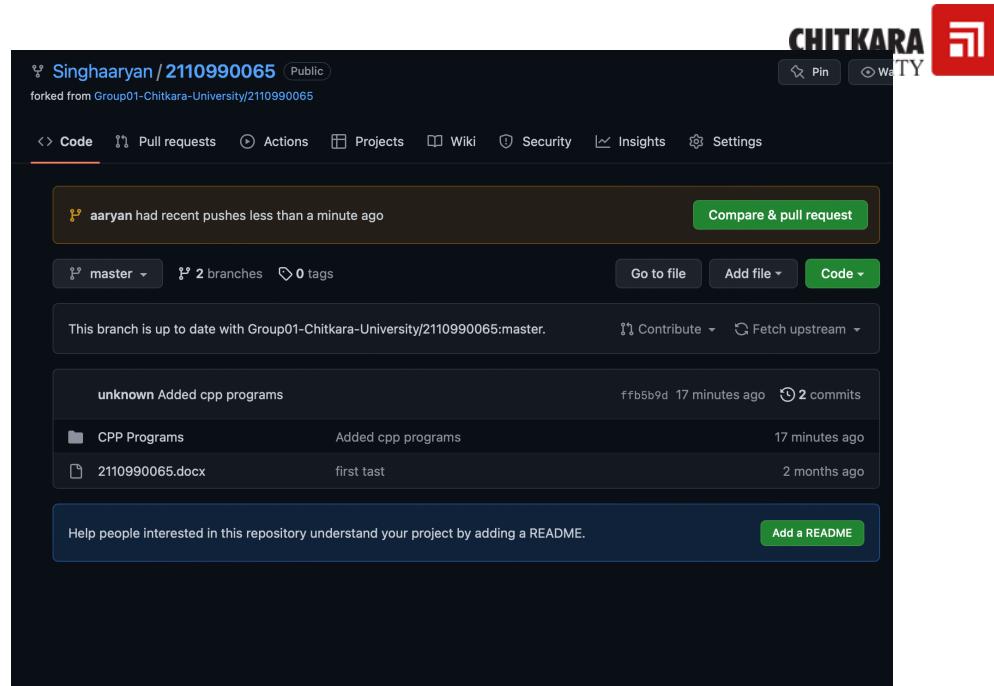
senpai@Aaryans-MacBook-Air 2110990065 % git commit -m "Add Files in program folder"
[aaryan c95636c] Add Files in program folder
Committer: Aaryan Singh <senpai@Aaryans-MacBook-Air.local>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly:
git config --global user.name "Your Name"
git config --global user.email you@example.com

After doing this, you may fix the identity used for this commit with:
git commit --amend --reset-author

2 files changed, 0 insertions(+), 0 deletions(-)
create mode 100644 .DS_Store
create mode 100755 CPP_Programs/exceptionhandling
senpai@Aaryans-MacBook-Air 2110990065 % git push origin aaryan
Username for 'https://github.com': Singhaaryan
Password for 'https://Singhaaryan@github.com':
remote: Support for password authentication was removed on August 13, 2021. Please use a personal access token instead.
remote: Please see https://github.blog/2020-12-15-token-authentication-requirements-for-git-operations/ for more information.
fatal: Authentication failed for 'https://github.com/Singhaaryan/2110990065.git'
senpai@Aaryans-MacBook-Air 2110990065 % $ git config --global user.name
zsh: command not found: $
senpai@Aaryans-MacBook-Air 2110990065 % git config --global user.name
senpai@Aaryans-MacBook-Air 2110990065 % git config --global user.name "Singhaaryan"
senpai@Aaryans-MacBook-Air 2110990065 % git push origin aaryan
Username for 'https://github.com': Singhaaryan
Password for 'https://Singhaaryan@github.com':
remote: Support for password authentication was removed on August 13, 2021. Please use a personal access token instead.
remote: Please see https://github.blog/2020-12-15-token-authentication-requirements-for-git-operations/ for more information.
fatal: Authentication failed for 'https://github.com/Singhaaryan/2110990065.git'
senpai@Aaryans-MacBook-Air 2110990065 % $ git push origin aaryan
Username for 'https://github.com': Singhaaryan
Password for 'https://Singhaaryan@github.com':
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 8 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 5.42 KiB | 5.42 MiB/s, done.
Total 5 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 1 local object.
remote:
remote: Create a pull request for 'aaryan' on GitHub by visiting:
remote:   https://github.com/Singhaaryan/2110990065/pull/new/aaryan
remote:
To https://github.com/Singhaaryan/2110990065.git
 * [new branch]    aaryan -> aaryan
senpai@Aaryans-MacBook-Air 2110990065 %

```

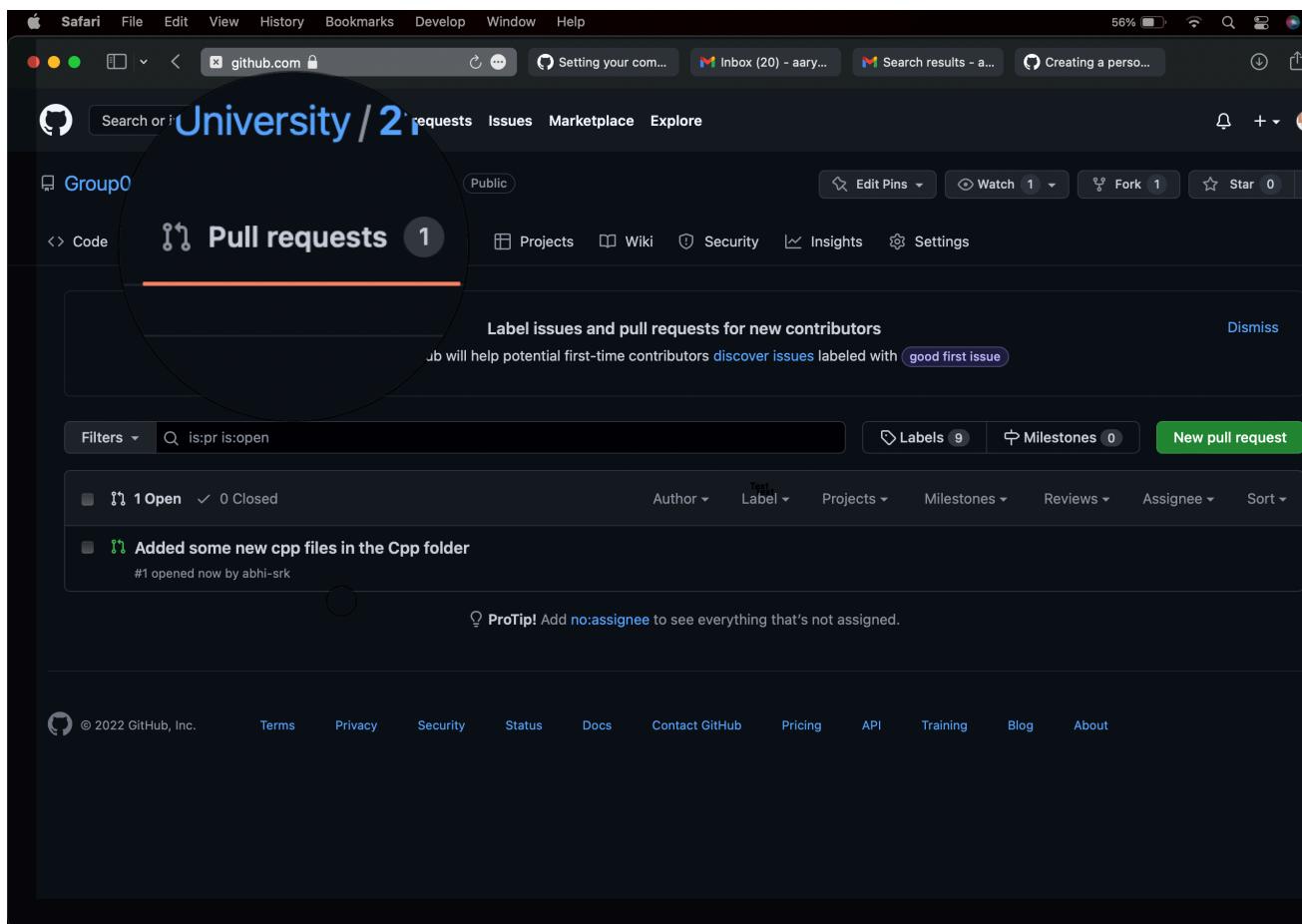
- After pushing new branch GitHub will either automatically ask you to create a pull request or you can create your own pull request.
- To create your own pull request click on pull request option.
- GitHub will detect any conflicts and ask you to enter a description of your pull request.



- 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.

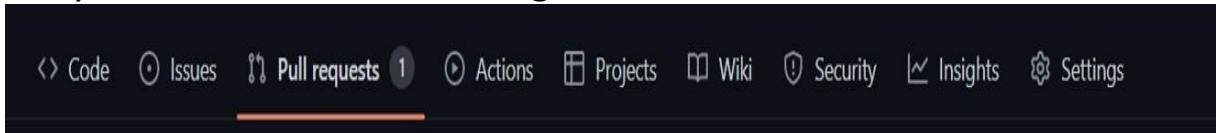


•

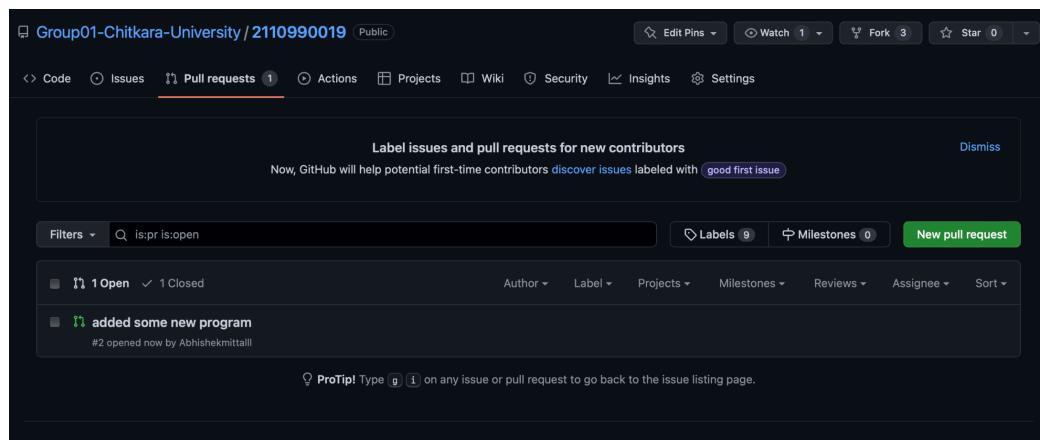
Aim: Create a pull request on a team member's repo and close pull requests generated by team members on own Repo as a maintainer

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:-

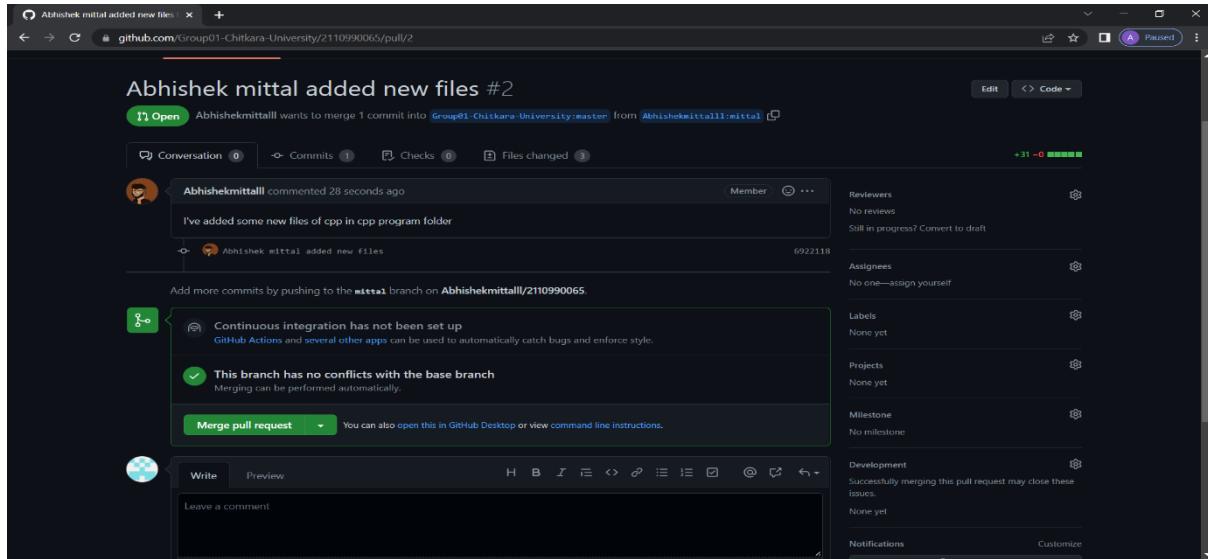
- Do the required changes in the repository, add and commit these changes in the local repository in a new branch.
- Push the modified branch using git push origin branchname.
- Open a pull request by following the procedure from the above experiment.
- The pull request will be created and will be visible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



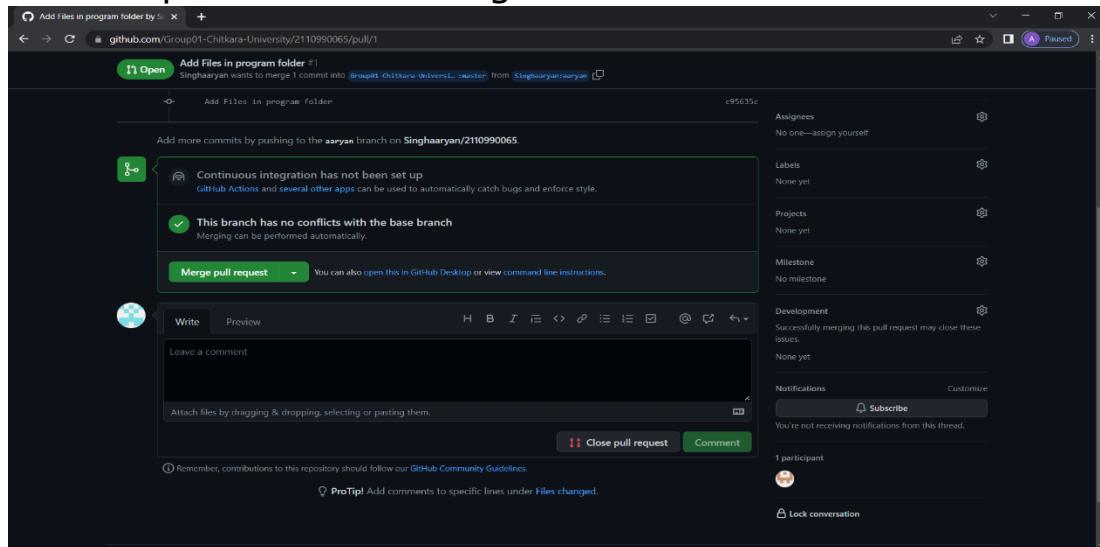
- They will notice a new notification in the pull request menu.
- Click on it. The pull request generated by you will be visible to them.



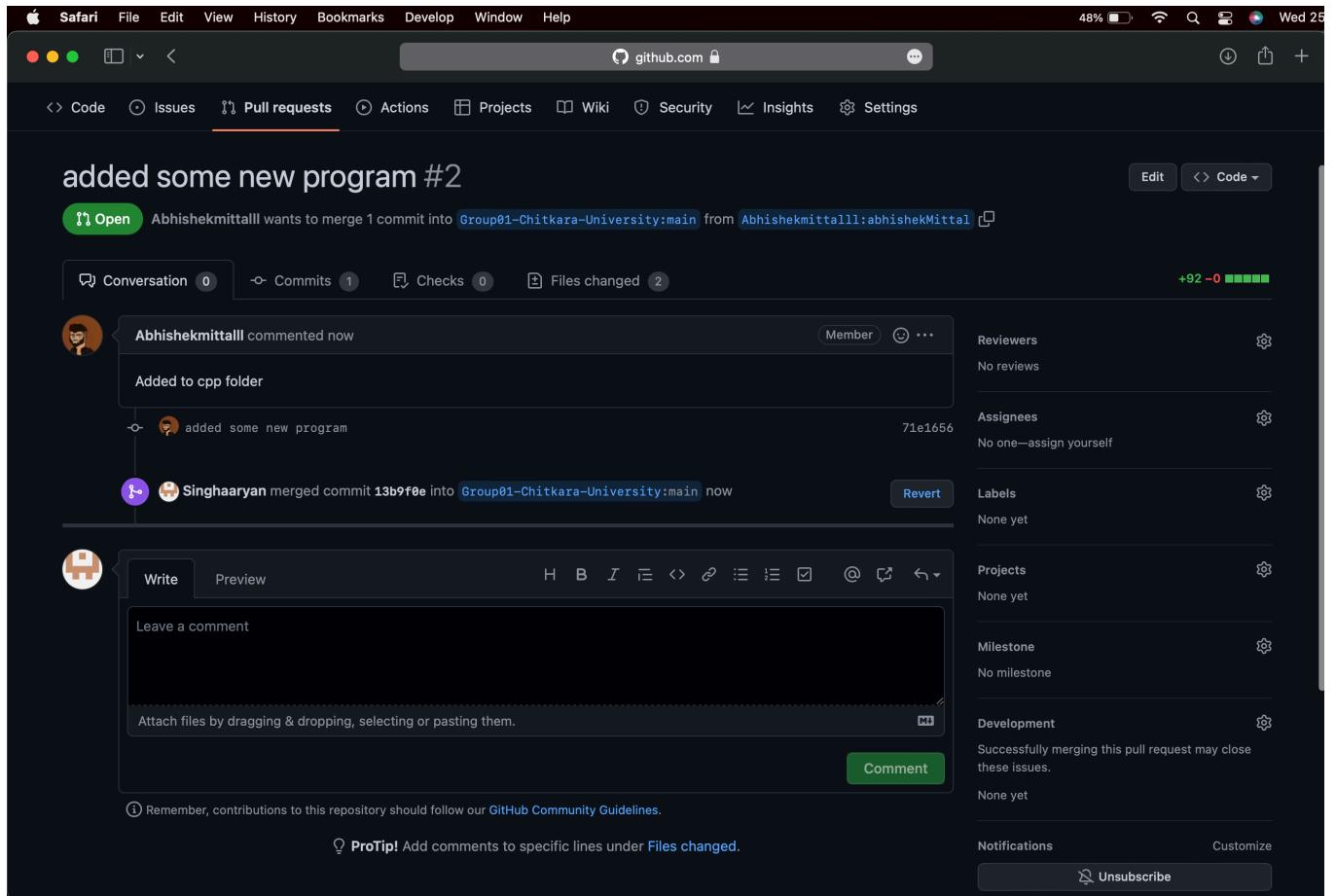
- Click on the pull request. Two option will be available, either to close the pull request or Merge the request with the main branch.
- By selecting the merge branch option the main branch will get updated for all the team members.



- 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.
- The result of closing the request is shown below.



The screenshot shows a GitHub pull request page for a repository named "Group01-Chitkara-University:main". The pull request has been merged by a user named "Singhaaryan" into the main branch. The commit message is "added some new program #2". A comment from "Abhishekmittalll" says "Added to cpp folder". Another comment from "Singhaaryan" says "added some new program". The pull request has 1 commit, 0 checks, and 2 files changed. The GitHub interface includes tabs for Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. On the right side, there are sections for Reviewers, Assignees, Labels, Projects, Milestone, Development, and Notifications.

- Thus, we conclude opening and closing of pull request. We also conclude merging of the pull request to the main branch.

Aim: Publish and print network graphs

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

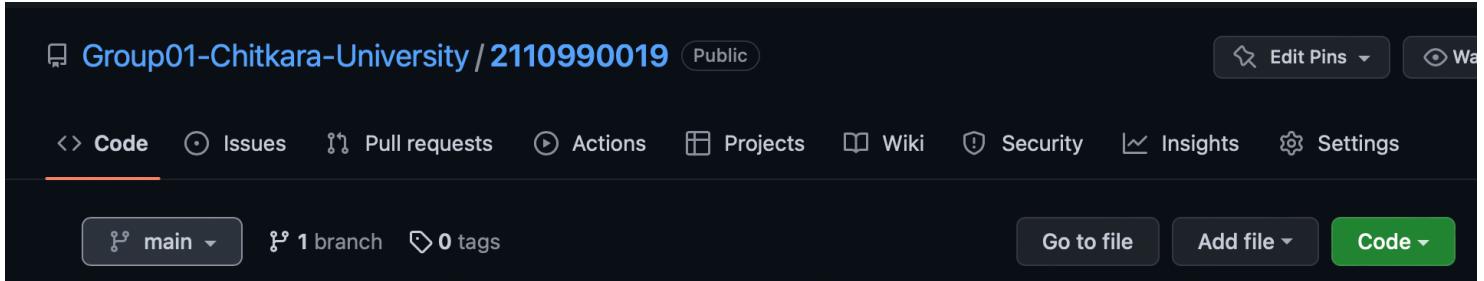
A repository's graphs give you information on traffic, projects that depend on the repository, contributors and commits to the repository, and a repository's forks and network. If you maintain a repository, you can use this data to get a better understanding of who's using your repository and why they're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

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

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

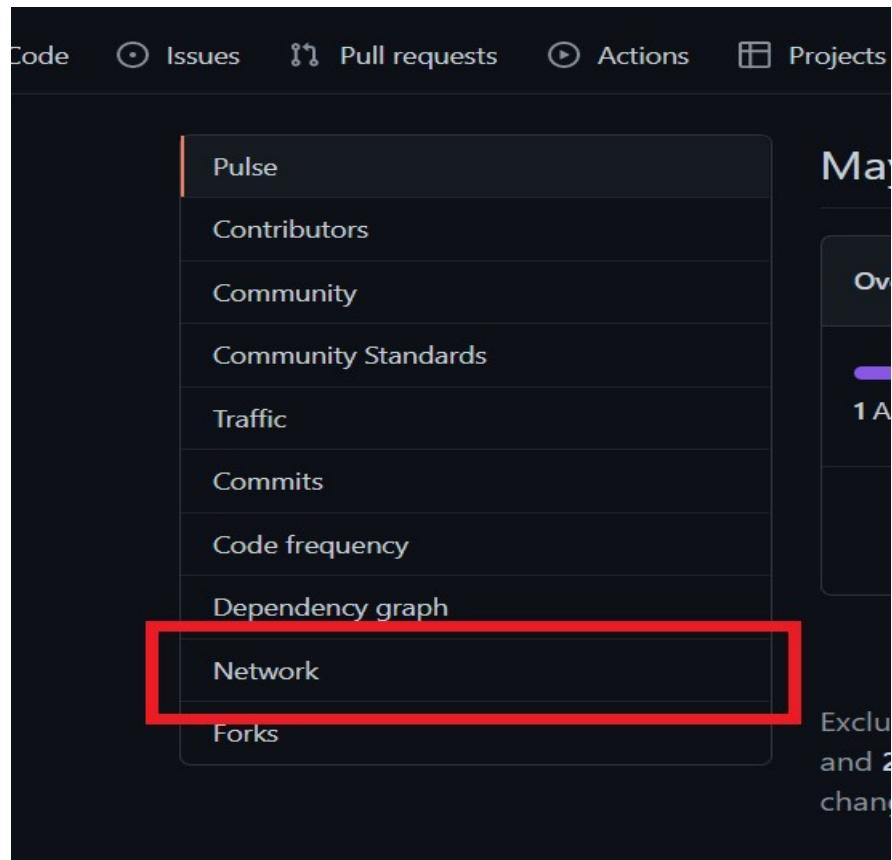
Steps to access network graphs of respective repository

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

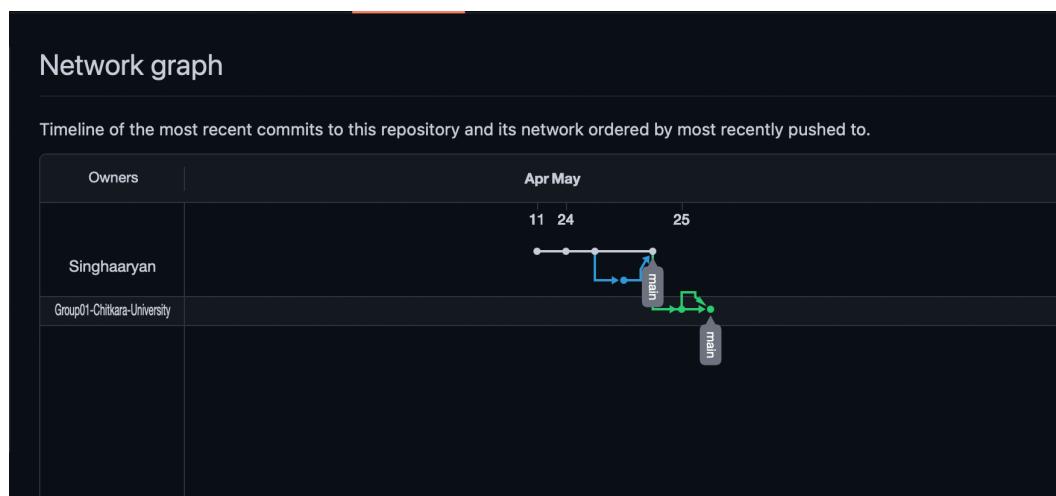


The screenshot shows the main interface of a GitHub repository. At the top, the repository name "Group01-Chitkara-University / 2110990019" is displayed with a "Public" badge. To the right are buttons for "Edit Pins" and "Watch". Below the repository name, a navigation bar includes links for "Code", "Issues", "Pull requests", "Actions", "Projects", "Wiki", "Security", "Insights" (which is highlighted in blue), and "Settings". Underneath the navigation bar, there are status indicators: "main" branch, "1 branch", "0 tags", "Go to file", "Add file", and a "Code" button. The background of the page is dark.

3. At the left sidebar, click on Network.



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

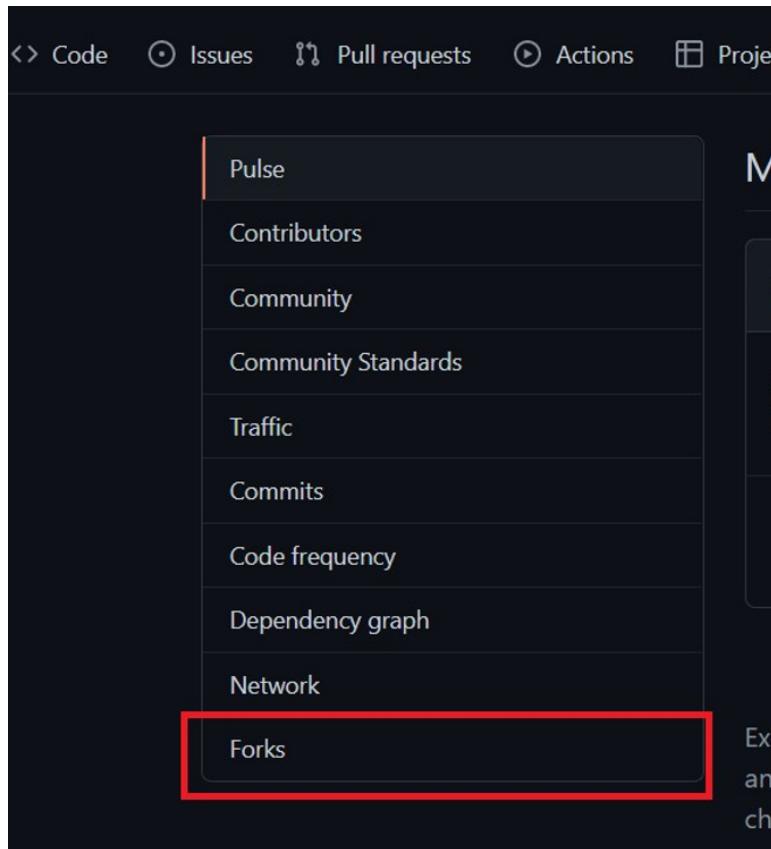


Listing the forks of a repository

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.



3. In the left sidebar, click Forks.

- 4.



Here you can see all the forks

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.

S.No.	

1	https://www.geeksforgeeks.org/version-control-systems/
2	https://www.perforce.com/blog/vcs/what-is-version-control
3	https://medium.com/@lanceharvieruntime/ version-control- why-do-we-need-it- 1681f4888cec
4	https://en.wikipedia.org/wiki/Git
5	https://git-scm.com/
6	https://www.simplilearn.com/tutorials/git-tutorial/git-vs-github