Subject Name: Source Code Management

Subject Code: 22CS003

Session: 2022-23

Department: DCSE

****

Submitted By: Submitted To:

Akshi Tripathi Dr. G.G.S. Pradeep

2210991233

G2-B(First Year)

List of Programs

|  |  |  |
| --- | --- | --- |
| S.NO. | Program 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 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 |  |

EXPERIMENT -1

Aim- Setting up of Git Client

Theory- Git is a distributed version control system that tracks changes in any set of computer files, usually used for coordinating work among programmers collaboratively developing source code during software development. Git and GitHub are used parallel to each other to edit, recreate, copy or download any code on github using git.

Procedure- We can download git from its official website using the link [https://git](https://git-scm.com/download/win)-[scm.com/download/win](https://git-scm.com/download/win) or we can just type scm git on google or any other search engine and we get directed to our required page. We can select our desired platform and bit-version and click on it and the download will start automatically.

Snapshots of downloading

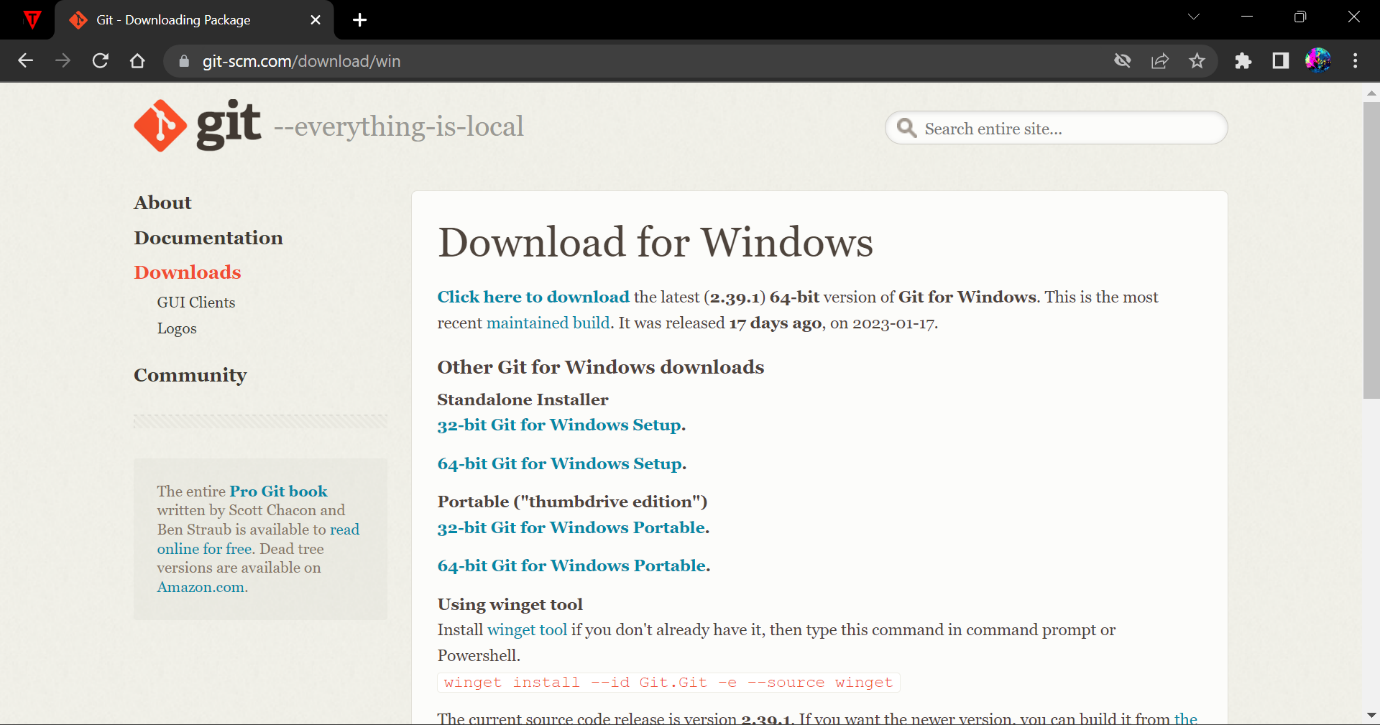


Fig.1 Opted for “64-bit Git for Windows Setup”

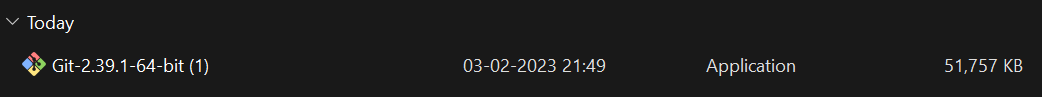


Fig.2 After downloading this git setup application appears in downloads of your pc.

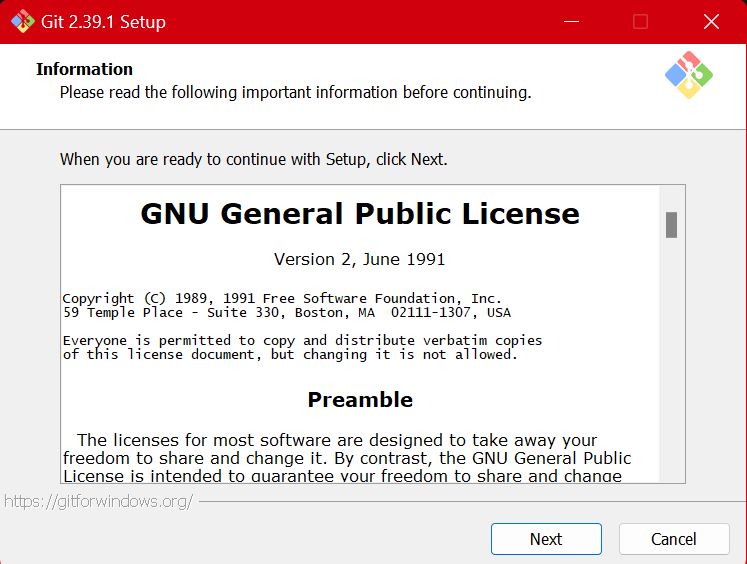


Fig.3 Git Setup

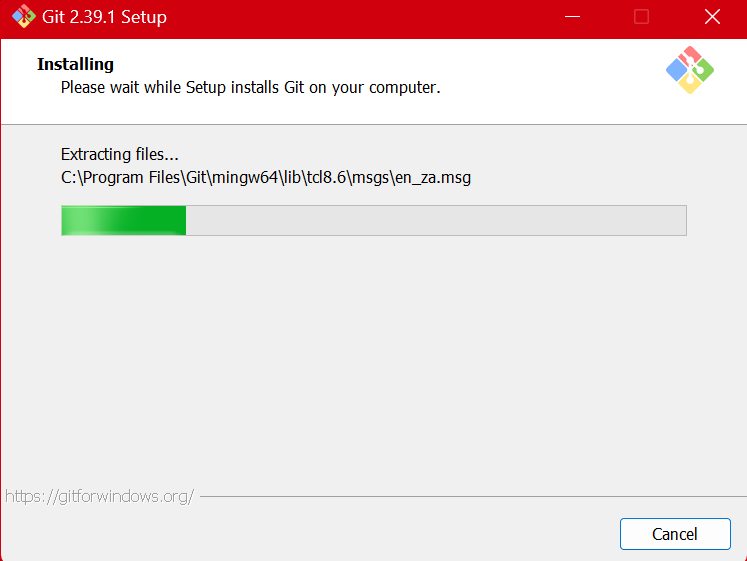


Fig.4 Git Installation

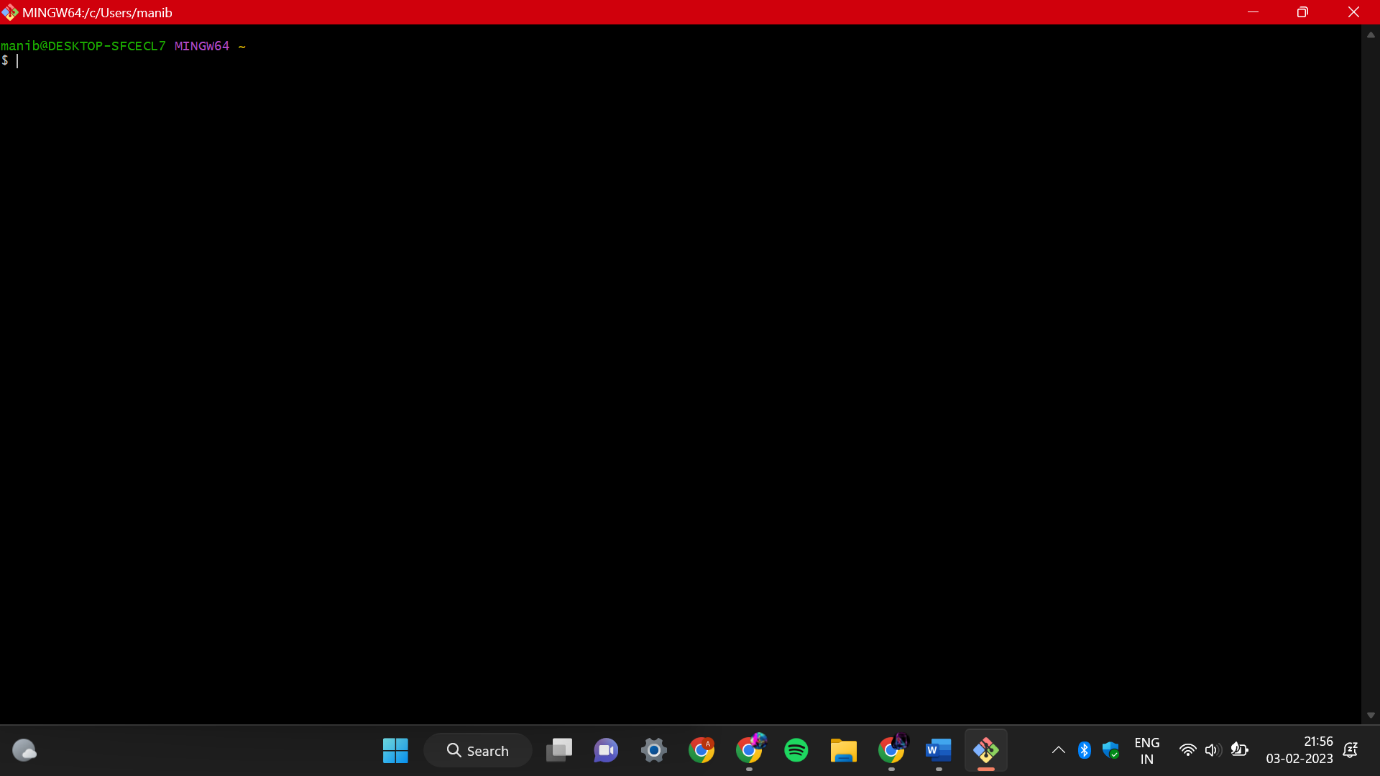


Fig.5 Git Bash launched

EXPERIMENT-2

Aim- Setting up GitHub Account

Theory- GitHub is an online software development platform. It’s used for storing, tracking, and collaborating on software projects.

Advantages of GitHub: GitHub has a user-friendly interface and is easy to use. 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 with 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- We can create a GitHub account by typing GitHub on our browser or just by clicking this link <https://github.com/signup> we would be directed to the home page. Then by entering your email id you can sign up or login by entering your username and password.

Snapshots of setting up account

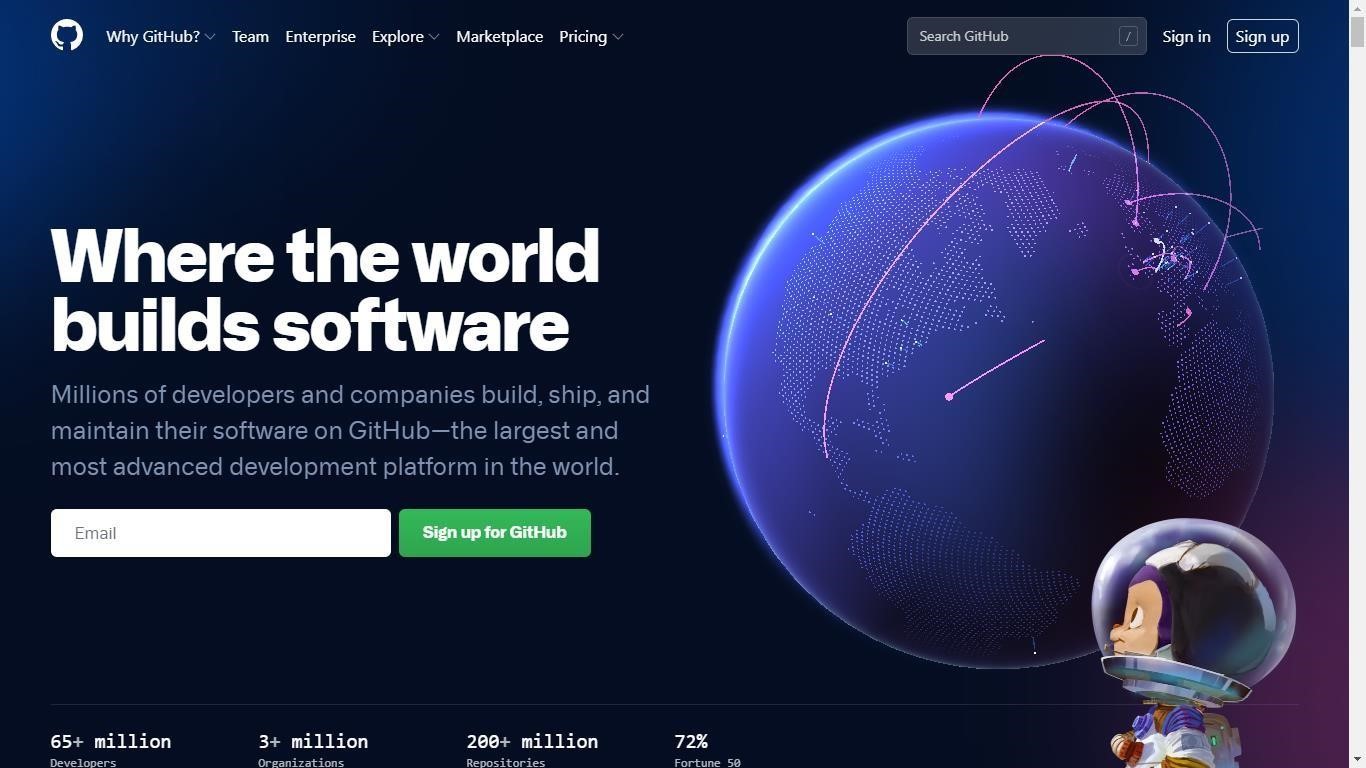


Fig1. After clicking on the link this page pops up and you can either login if you have an id or create one.

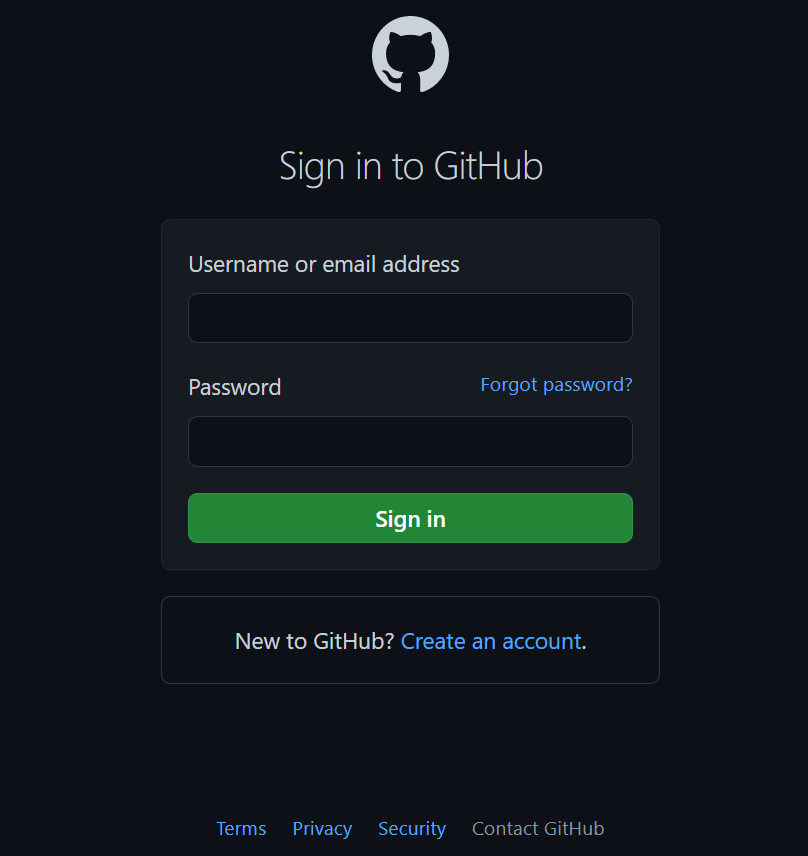


Fig.2 GitHub login

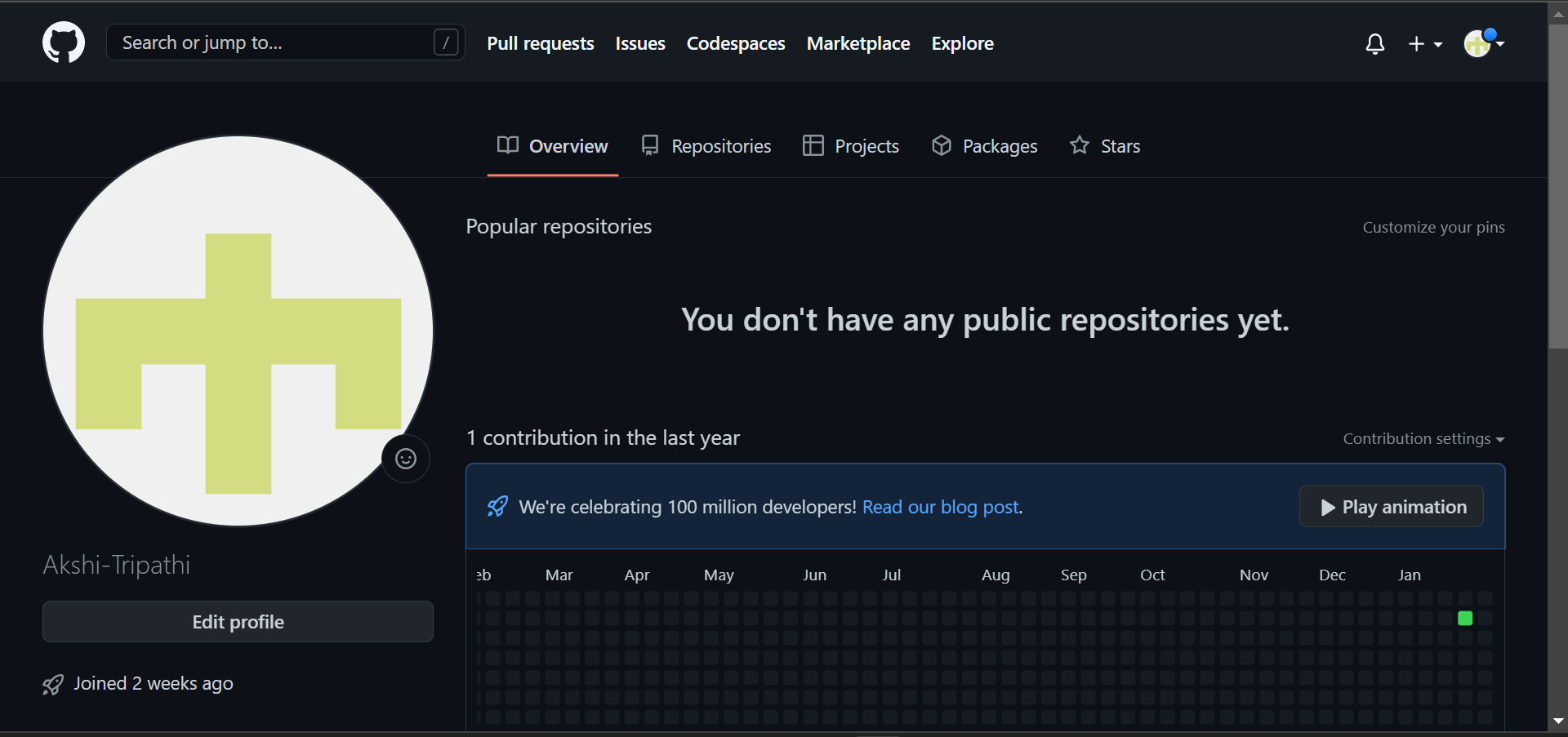


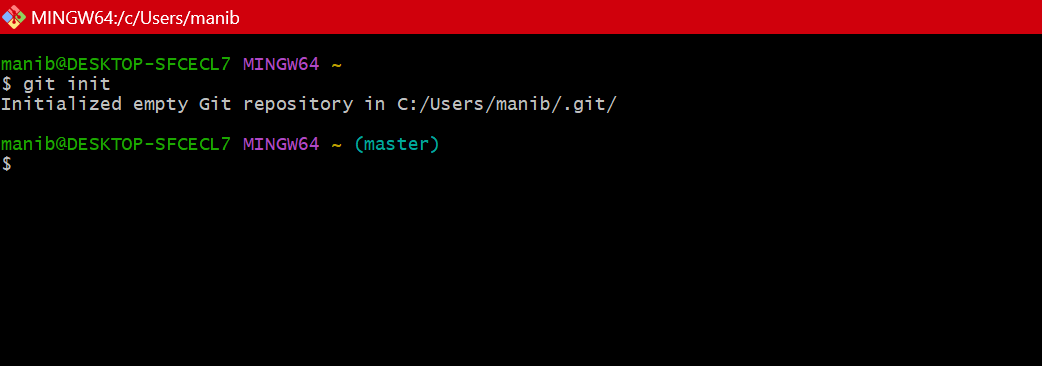
Fig3. GitHub Interface

EXPERIMENT-3

Aim- Program to generate logs

Theory- Logs are the history which we can be seen 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. It helps to check that what were the changes in the code or any other file and by whom. It also contains number of insertions and deletions included 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”.



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”

## **Some Important Commands**

ls ~It gives the file names in the folder.

ls -lart ~Gives the hidden files also.

git status ~Displays the state of the working directory and the staged snapshot.

touch filename ~This command creates a new file in the repository.

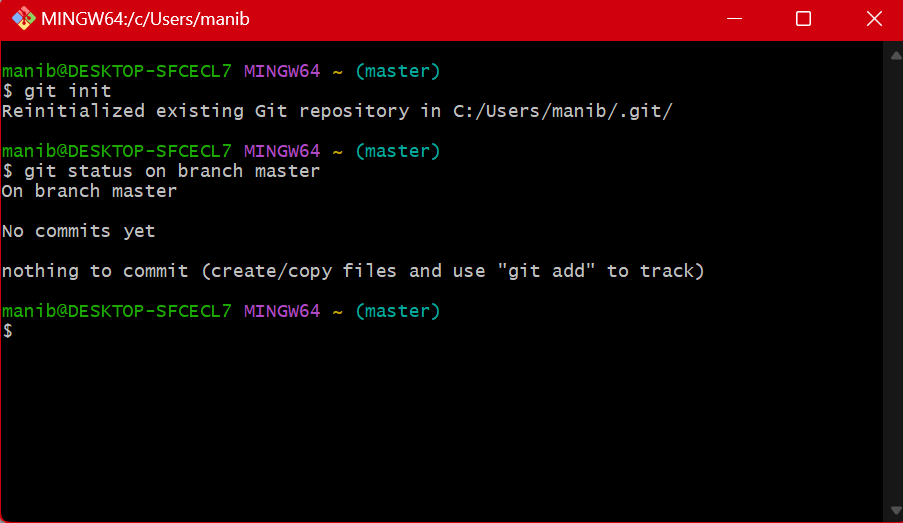
clear ~It clears the terminal.

rm -rf .git ~It removes the repository.

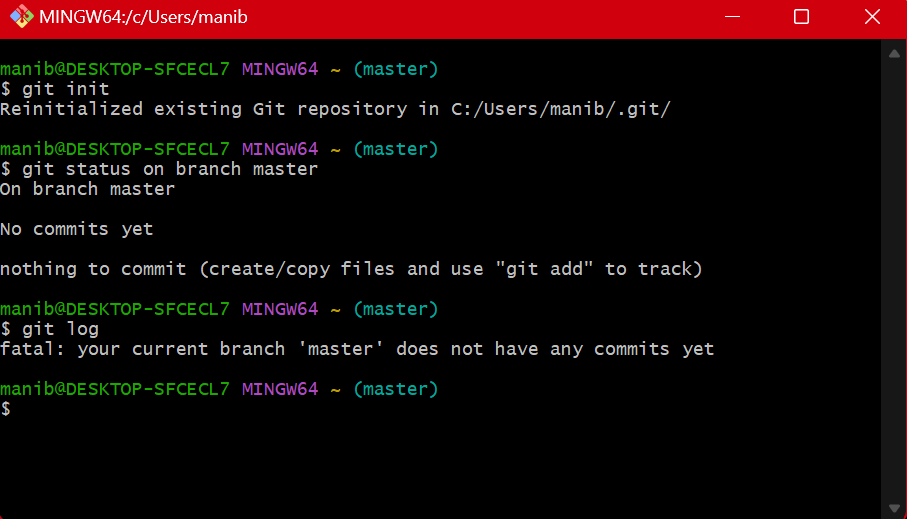
git log ~displays all of the commits in a repository's history.

git diff ~It compares my working tree to staging area.

Git status;



Git log;



EXPERIMENT-4

Aim- Create and visualize branches

Theory- A branch in git is simply a lightweight movable pointer to one of the commits. The default branch name in git is master. As you start making commits, you’re given a master branch that points to the last commit you made. Every time you commit, the master branch pointer moves forward automatically.

Creating a branch: The main branch in git is set as default to master. We can change the branch using the command “switch” or” checkout”.



Fig.1 Main branch “Master”



Fig.2 Using Checkout command

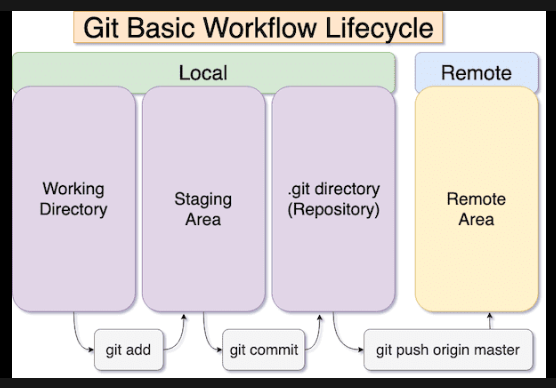


Fig. 3 Using switch command

EXPERIMENT-5

Aim- Git Lifecycle description

Theory- A code is copied from remote to local repository and then editing is done as per required and then moved in staging area. Further it is checked if any changes are made in remote repository by other users if yes hey are also pulled but if no then we push our changes to remote repository.



When a directory is made git repository then it mainly contains 4 parts:

1. Working directory:

Whenever we have to initialize a git repository we use “git init” command and then git still does not keep track of the files and they are tracked in staging area.

1. Staging area:

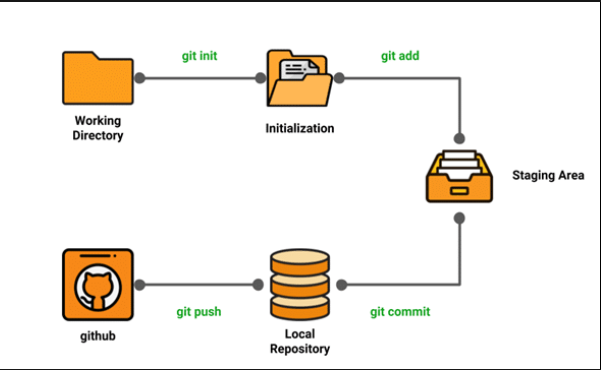
“Git add” command is used to keep track of all the files in this area.

1. 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.

1. 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

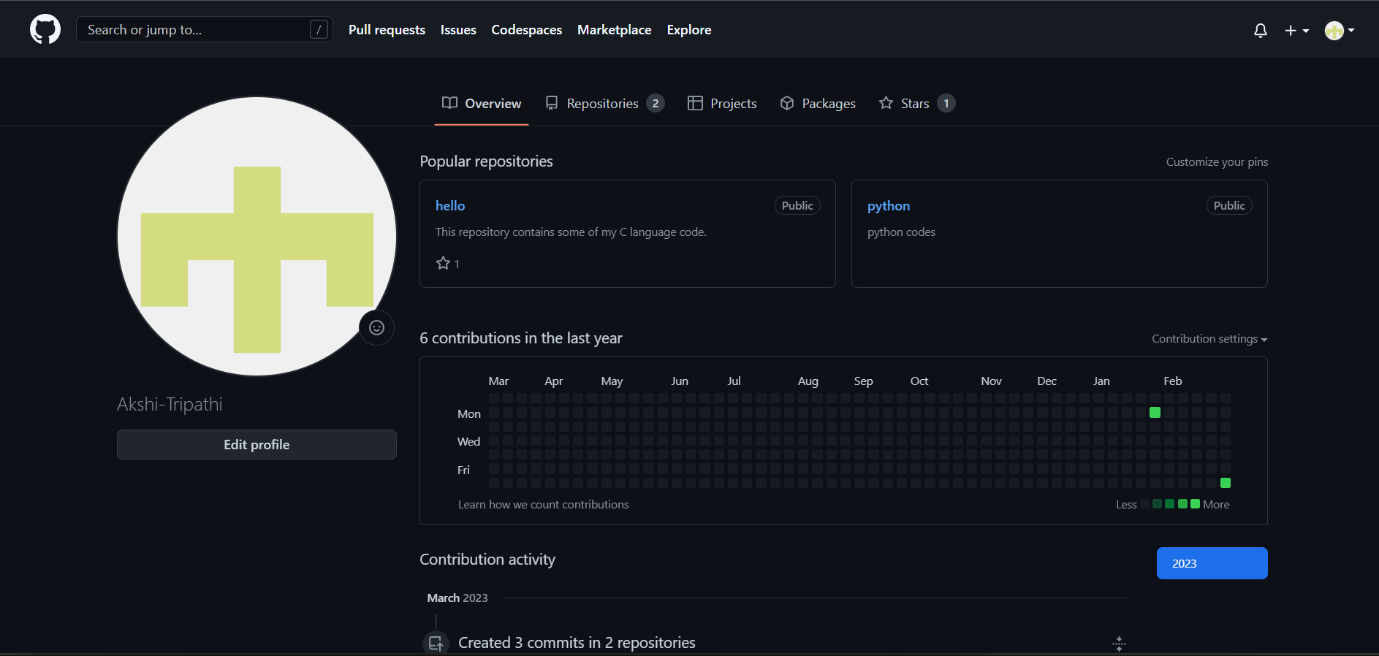
Theory- Whenever you make a repository in GitHub, not everyone has the permission to change or push codes into your repository. The users have a read-only access. In order to allow other individuals to make changes to your repository, you need to invite them to collaborate to the project.

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.

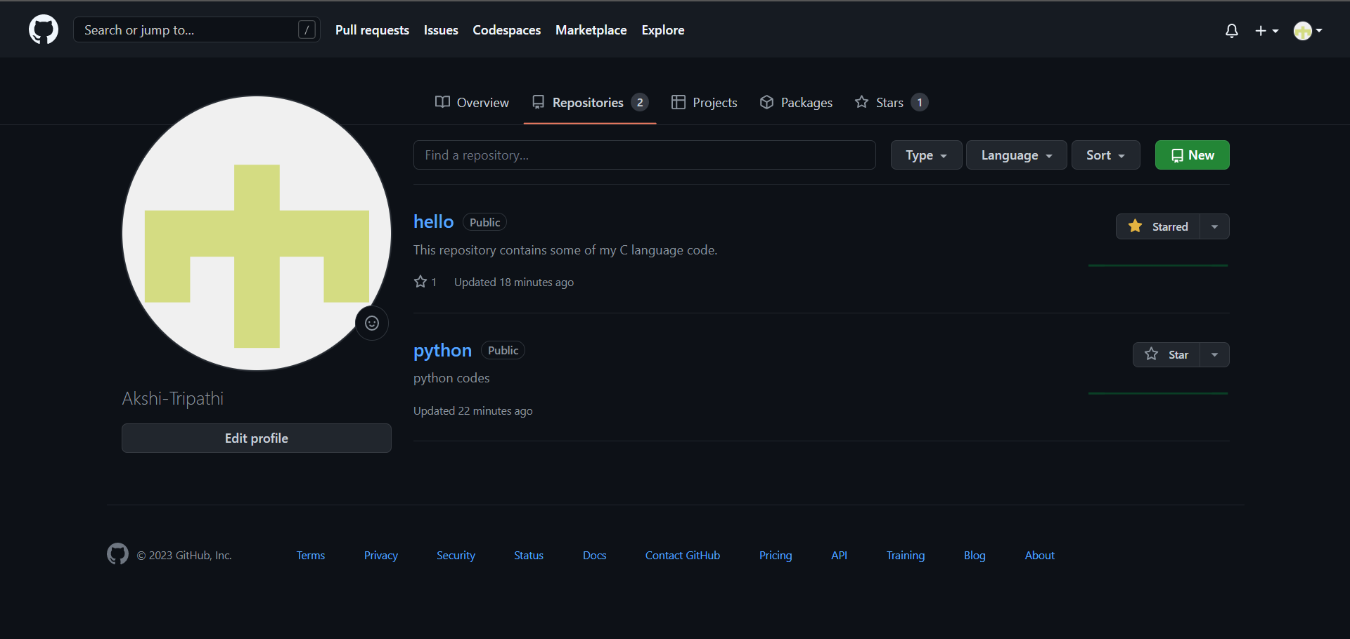
Being a collaborator, the user can create, merge and close pull requests in the repository. They can also remove them as the collaborator.

Procedure-

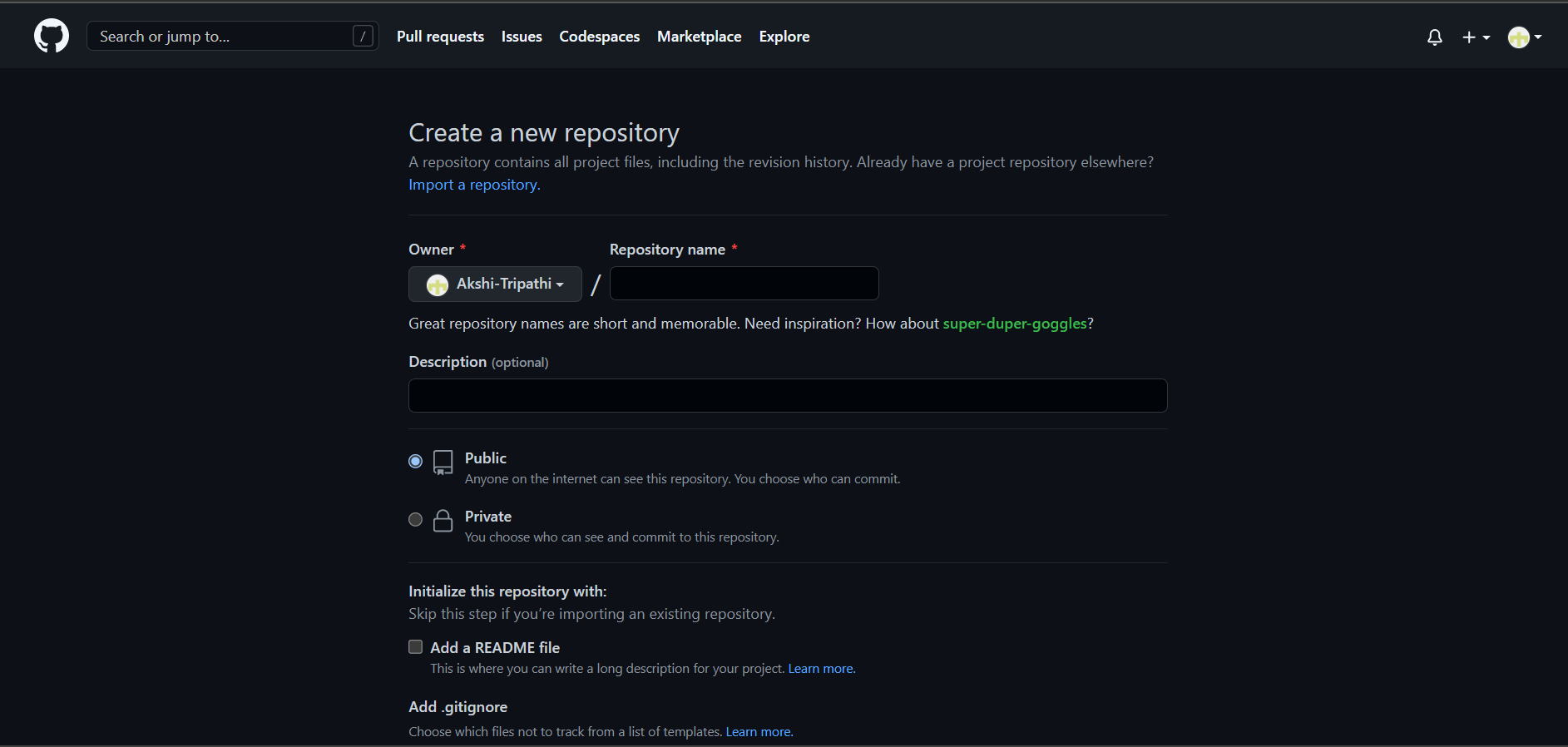
1. Login to GitHub account and open your account.



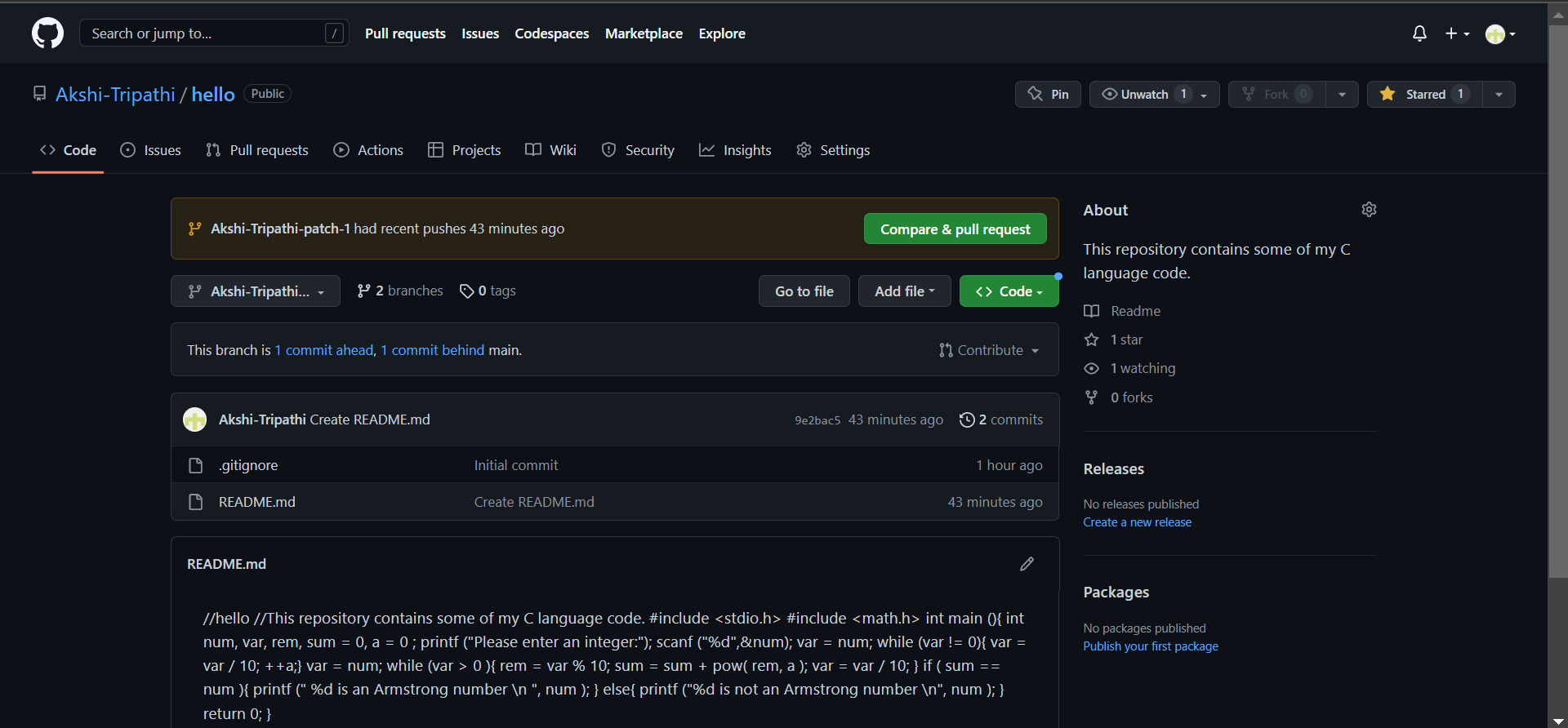
1. Click on the repositories option in the menu bar. To create a new repository click on the new option on the top corner.



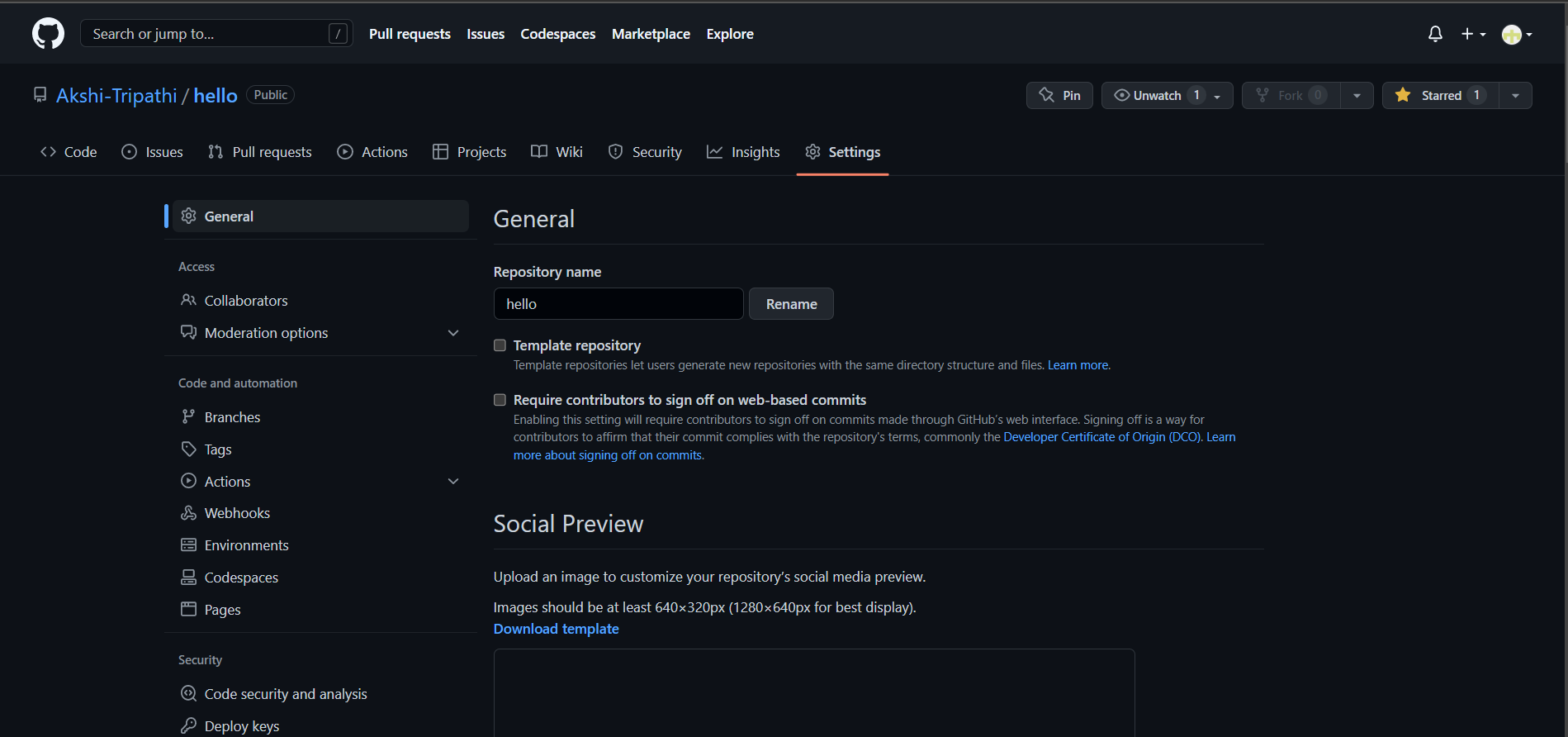
1. Add title and description of the repository. Also select if you want the repository to be private or public.



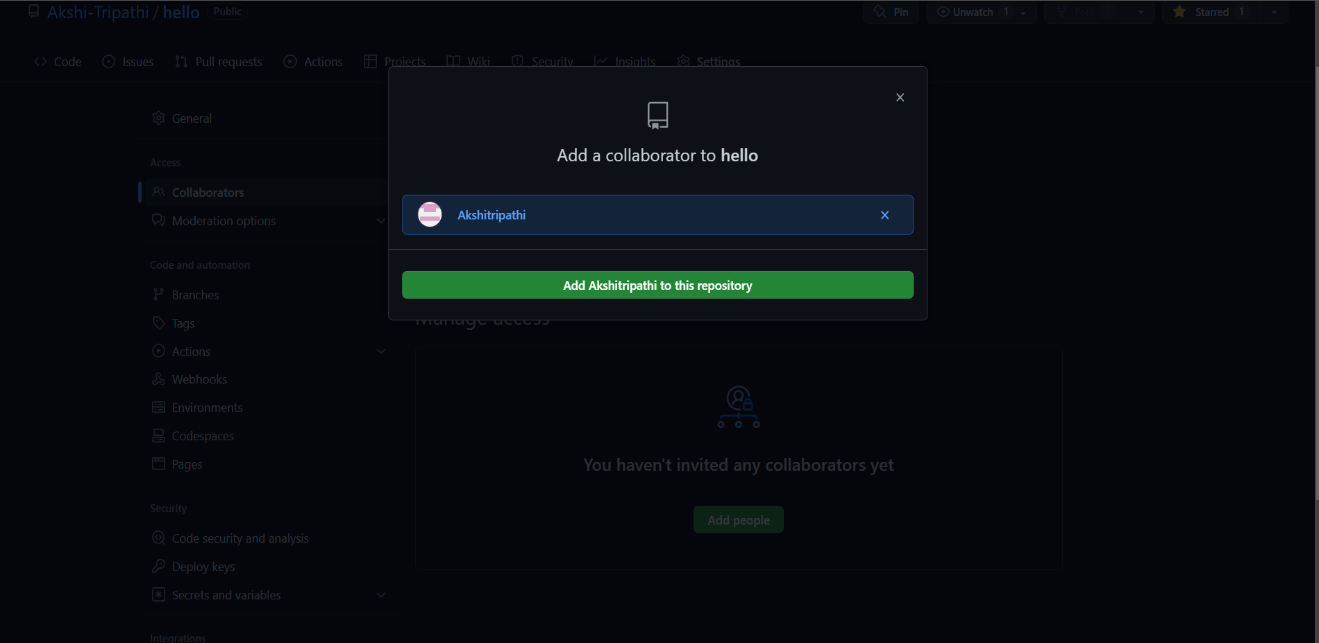
1. If you want to import code from some other repository you can import code and now you have created a repository successfully.
2. To add collaborators to your repository, open your repository and select settings option in the navigation bar.



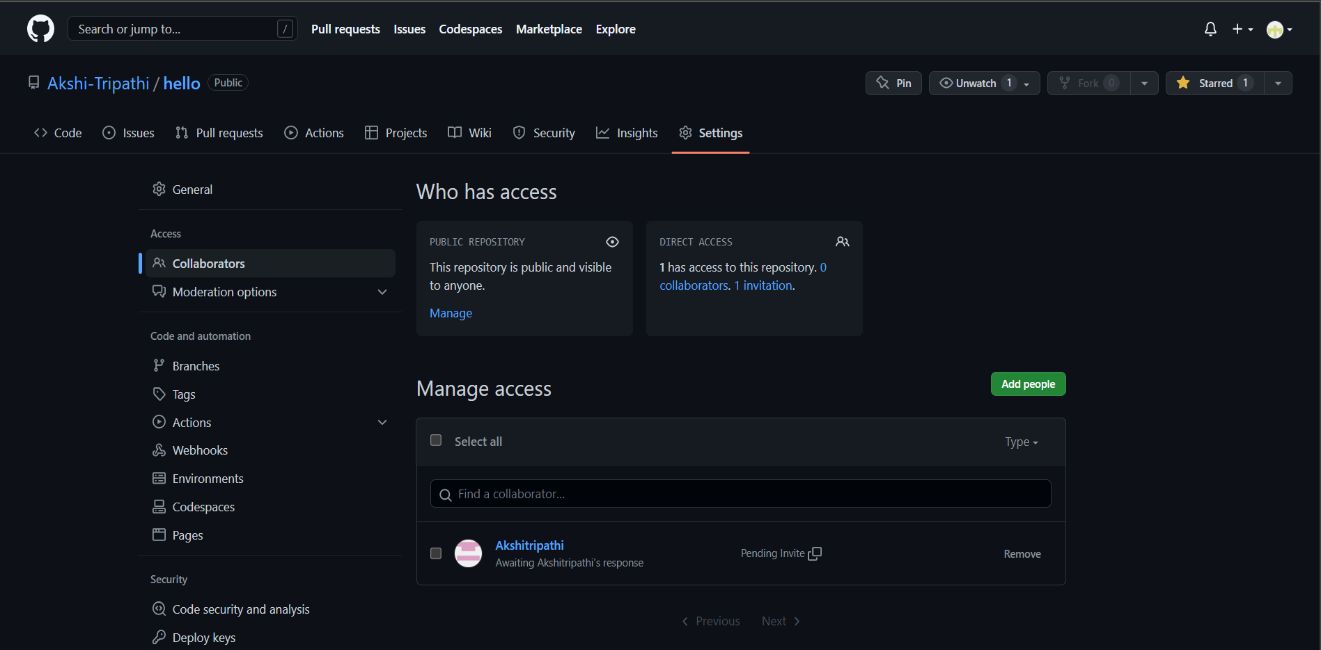
1. Click on Collaborators option under the access tab.



1. After clicking on collaborator GitHub asks you to verify your password. Once you verify the password now you can add/remove peoples having access to your repository.
2. To add members, click on add people and search id of the person.



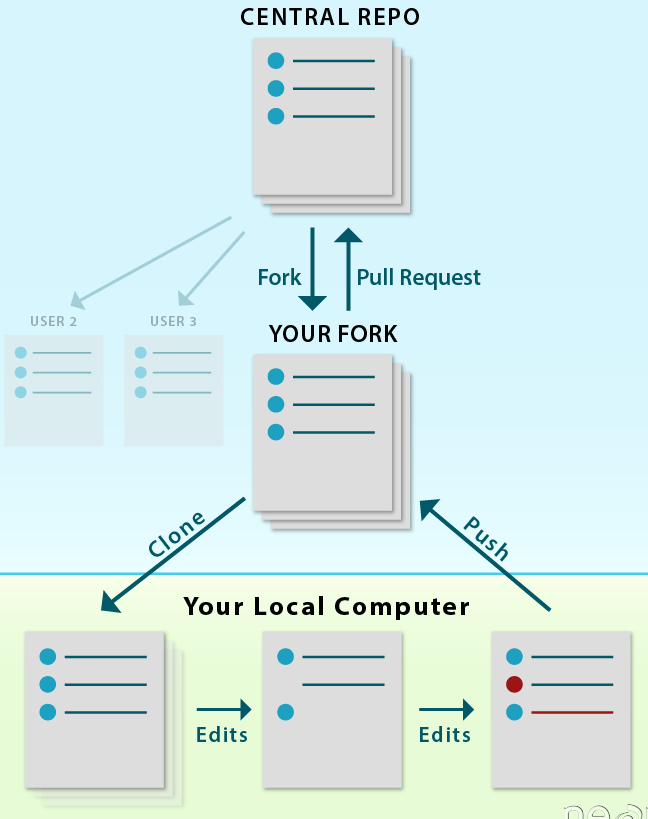
1. To remove person from repository you can click on remove on front of each member in respective row.



EXPERIMENT-7

Aim- Forks 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.

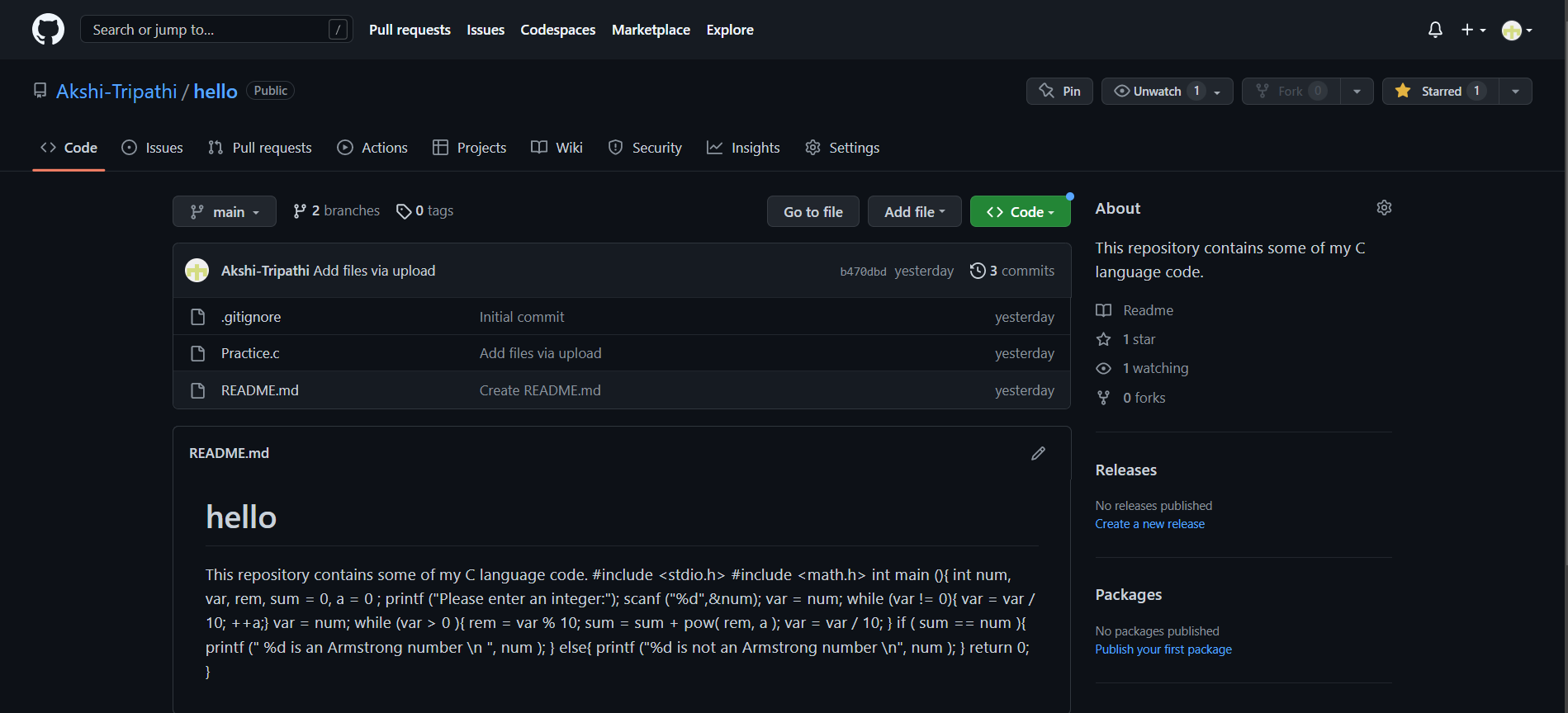


To import the changes into the original repository, the user needs to send a pull request to the maintainer. If the maintainer closes the pull request only then the content can be added to the original repository.

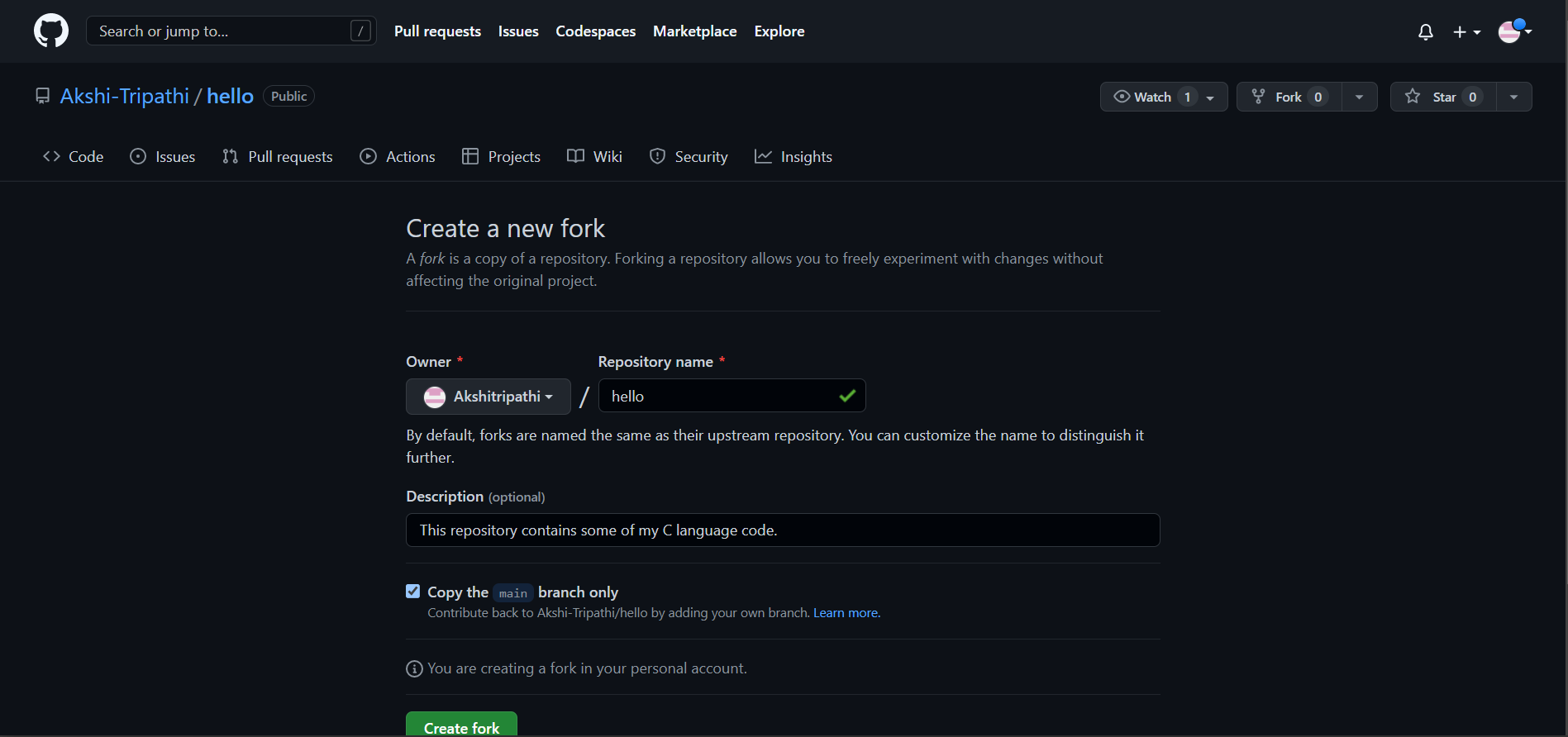
Forking is a better method than directly cloning any repository, as in cloning only the default branch is cloned whereas forking creates a clone of the complete repo and also allows us to push the changes to the main repository by using open and close pull request.

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 I am using Akshi-Tripathi/hello repository.

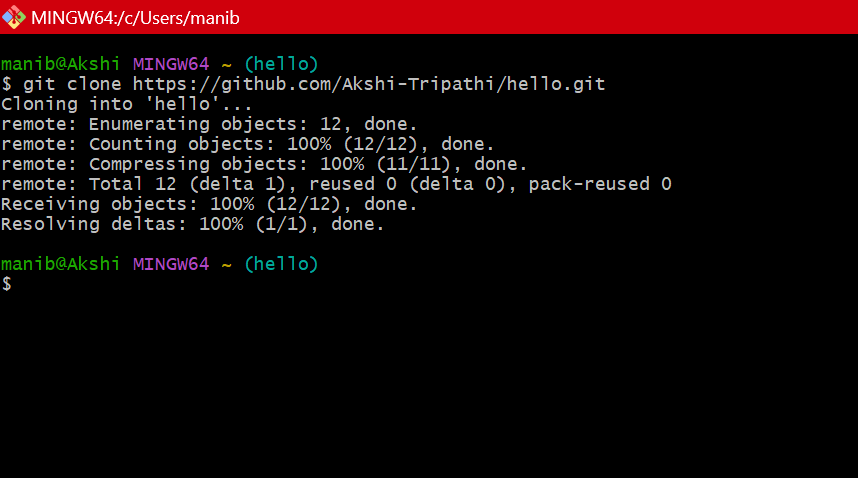


1. Click on the **Fork** button on right upside corner. Then it will ask to create a new fork, add description if you want and then click on create fork.

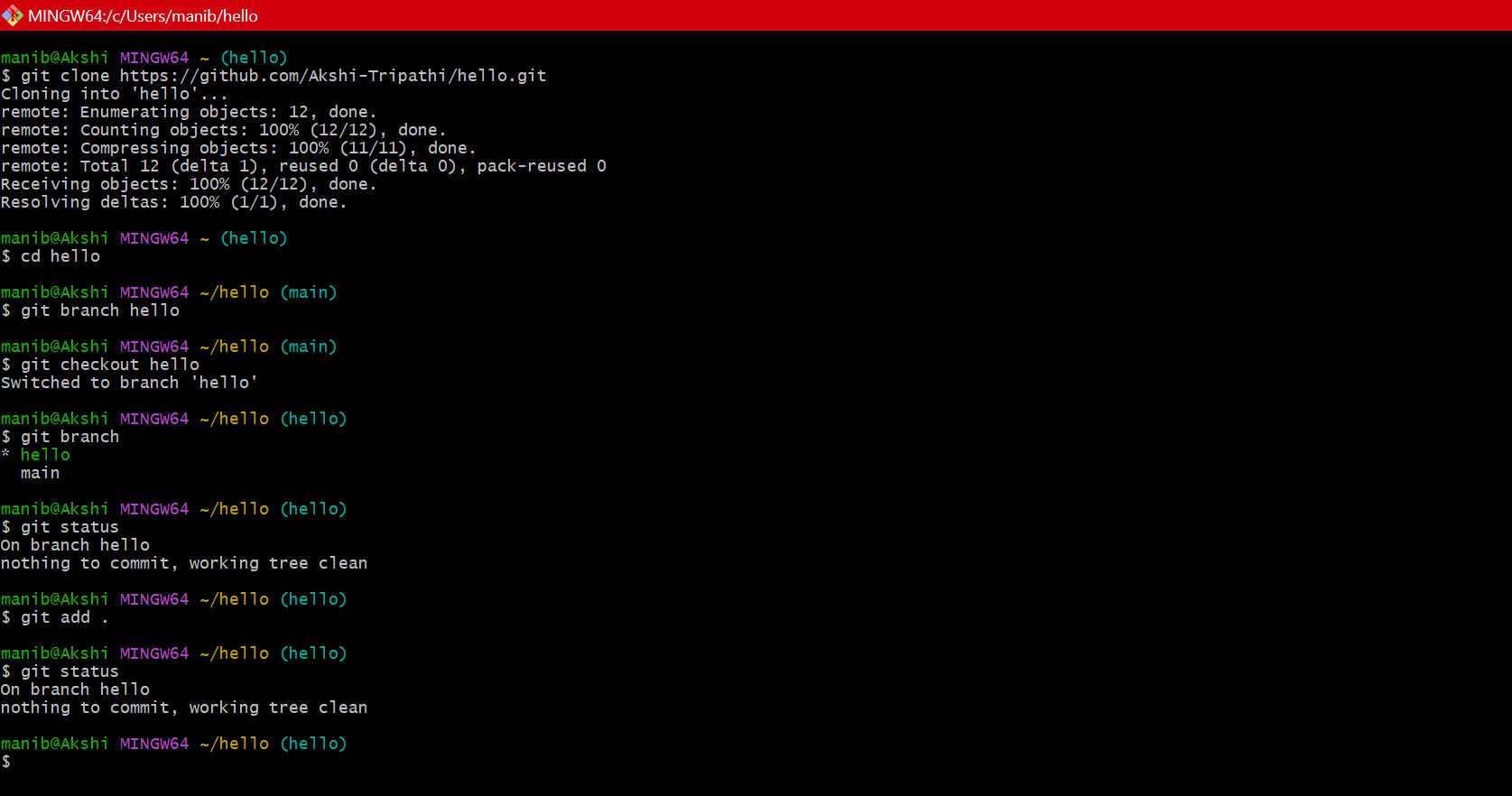


1. Now you will have a copy of the repo you have forked from other user. Now you can do any modification you want without making changes to main source code.
2. Now type git clone <https://github.com/Akshi-Tripathi/hello.git> on git.

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



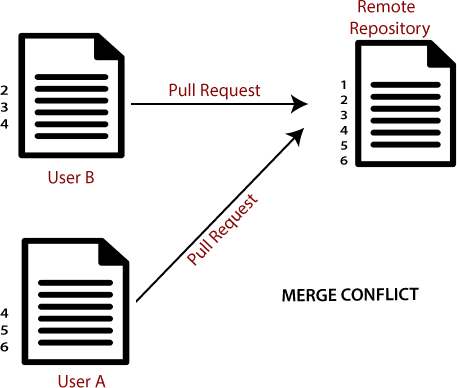
1. Now Open the file make changes in it and commit it and push it to remote.



EXPERIMENT-8

Aim- Merge and Resolve conflicts created due to own activity and collaborators activity

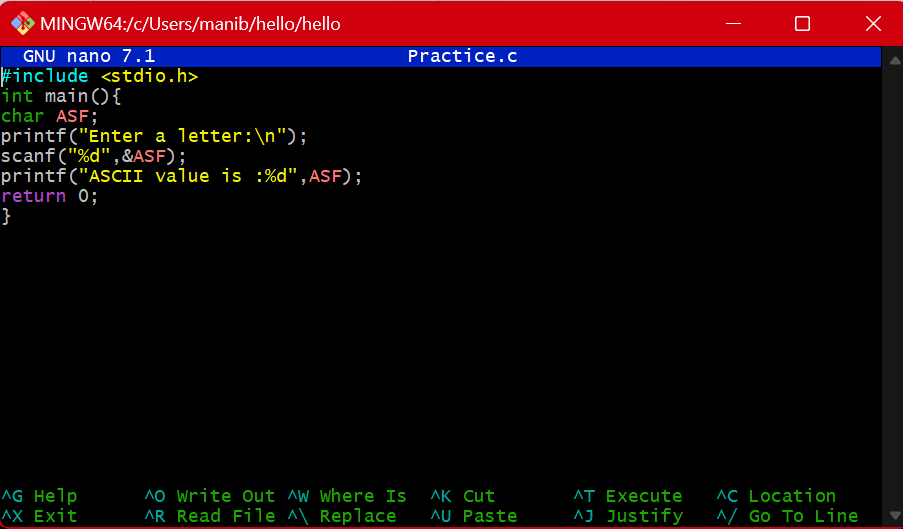
Theory- Version control systems are all about managing contributions between multiple distributed authors (usually developers). Sometimes multiple developers may try to edit the same content. If Developer A tries to edit code that Developer B is editing a conflict may occur.

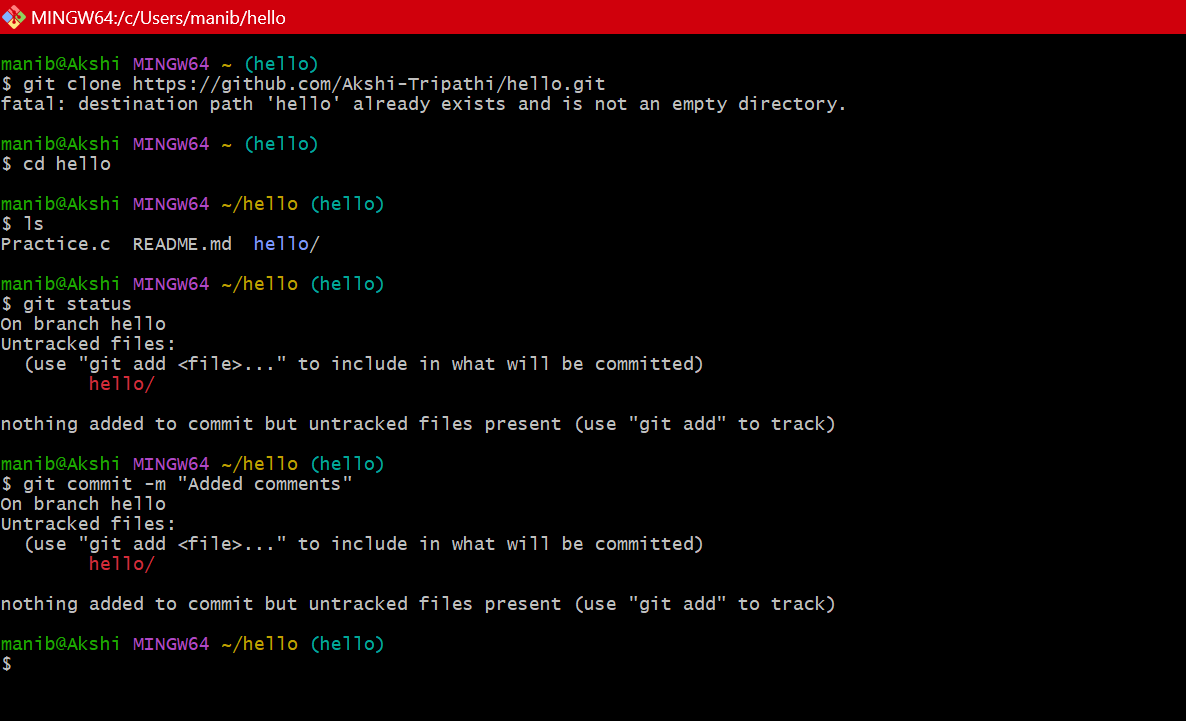


If you have a merge conflict on the command line, you cannot push your local changes to GitHub until you resolve the merge conflict locally on your computer. To alleviate the occurrence of conflicts developers will work in separate [isolated branches](https://www.atlassian.com/git/tutorials/using-branches). 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 pull request button is deactivated until you've resolved all conflicts between the compare branch and base branch.

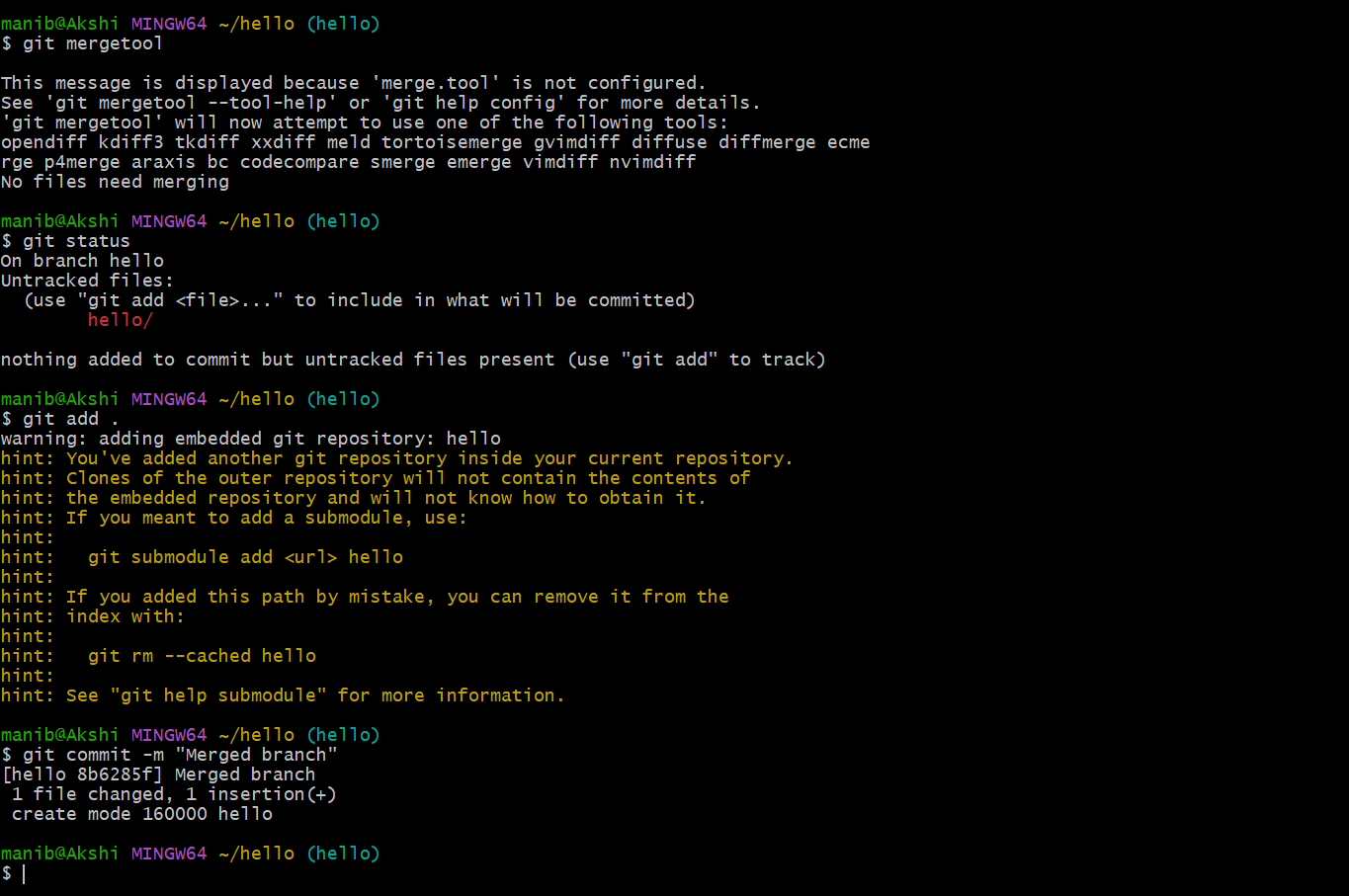
Procedure-

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. 
2. 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.

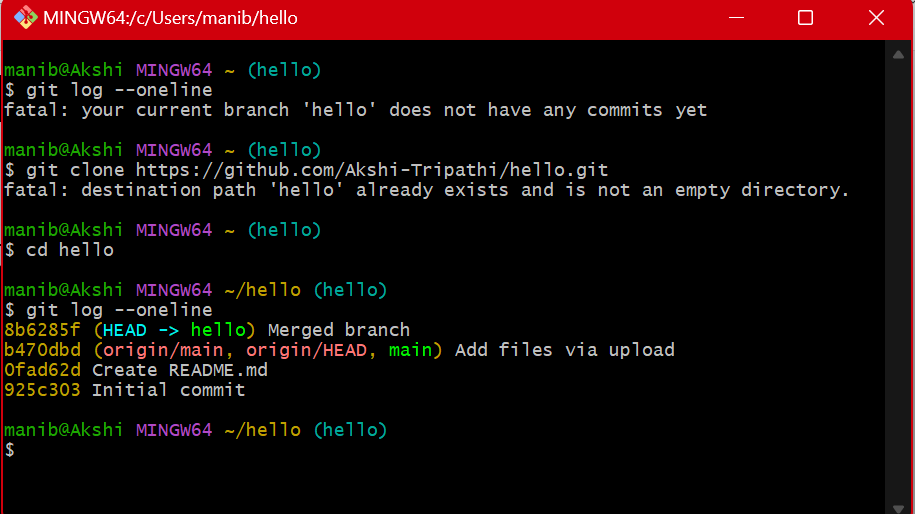
These two features justify the Version- controlled feature of the git as we can rollback to any version at any time.

### PROCEDURE:

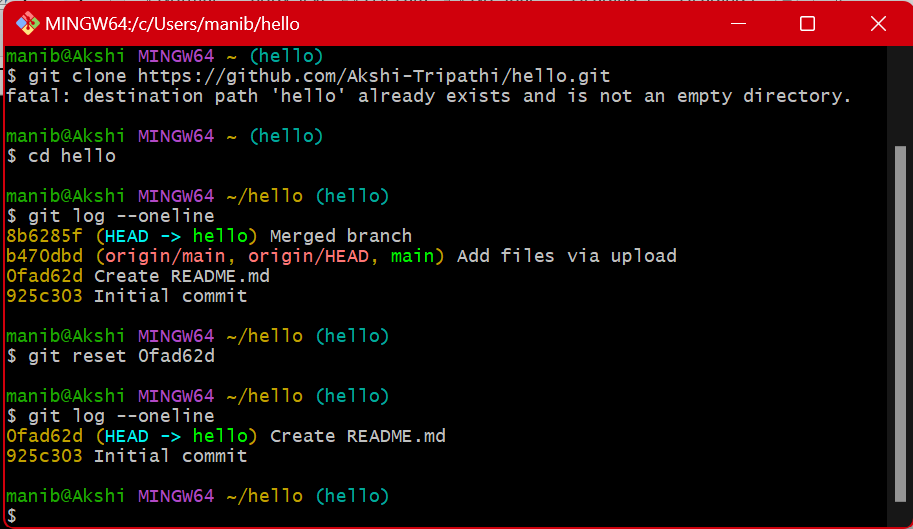
Firstly, prepare a log of multiple commits to make the reset and revert command function.

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

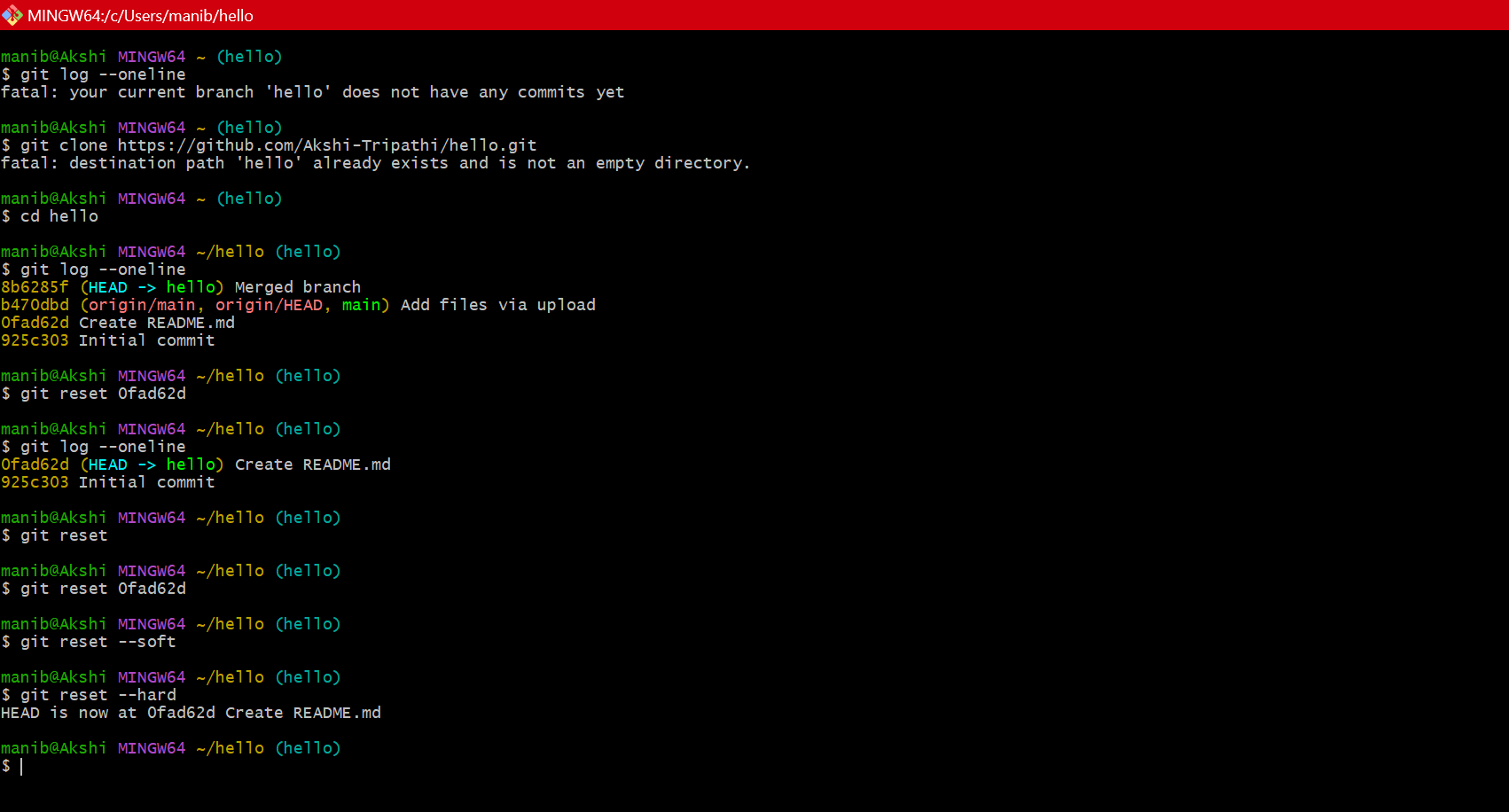


* 1. Pick any commit where you want the repository to rollback. Copy its checksum and paste it in the $ **git reset checksum** command.



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

* 1. 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.



* + 1. A window will appear. Press ‘I” and write the statement you want to be displayed for reverting the change.
    2. After completing press ‘esc’ and write: wq in the terminal.
    3. Check the git log and you will find another commit is added without affecting the rest commits.
    4. The change associated to the reverted commit has disappeared.