**Git & GitHub Document**

**Version: 2.45.0. windows.1**

**May 2024**

**Table of Contents**

[**1.** **Git and GitHub Introduction** 3](#_Toc166775836)

[**2.** **Creation of GitHub account** 4](#_Toc166775837)

[**2.1 GitHub Sign up** 4](#_Toc166775838)

[**3.** **Git Installation for Windows** 5](#_Toc166775839)

[**3.1 Verify the git installation** 5](#_Toc166775840)

[**4.** **Initialize a local git repository** 5](#_Toc166775841)

[**5.** **Creation of new Repository** 6](#_Toc166775842)

[**5.1 Login to GitHub** 6](#_Toc166775843)

[**5.2 Initialize with a README** 6](#_Toc166775844)

[**6.** **link our local repository to GitHub** 7](#_Toc166775845)

[**6.1 Get** **the Remote Repository URL** 7](#_Toc166775846)

[**6.2 Configure our Git username and mail** 8](#_Toc166775847)

[**7.** **Push the units to Remote repository** 8](#_Toc166775848)

[**8.** **Creation of branches in new repository** 10](#_Toc166775849)

[**8.1 Modifying name of Master to required one in new repository** 10](#_Toc166775850)

[**9.** **Clone a remote git Repository to our local Machine** 11](#_Toc166775851)

[**10.** **Modifying units from existing repository** 12](#_Toc166775852)

[**11.** **Merging of branches** 13](#_Toc166775853)

[**12.** **Merge Conflict** 14](#_Toc166775854)

[**13.** **Resolving a merge conflict in git involves a few steps** 15](#_Toc166775855)

**From Add to Merge: Git and GitHub Essentials**

# **Git and GitHub Introduction**

**What is Git**?

Git is a distributed version control system commonly used for tracking changes in source code during software development.

It allows multiple developers to work on the same project simultaneously and keeps track of all changes made to the codebase over time.

It is used for:

* Tracking code changes
* Tracking who made changes
* Coding collaboration

What does Git do?

* Manage projects with **Repositories**
* **Clone** a project to work on a local copy
* Control and track changes with **Staging** and **Committing**
* **Branch** and **Merge** to allow for work on different parts and versions of a project
* **Pull** the latest version of the project to a local copy
* **Push** local updates to the main project

What is GitHub?

GitHub is one of web-based hosting service for git, provides a "remote" location for storing our git workspaces.

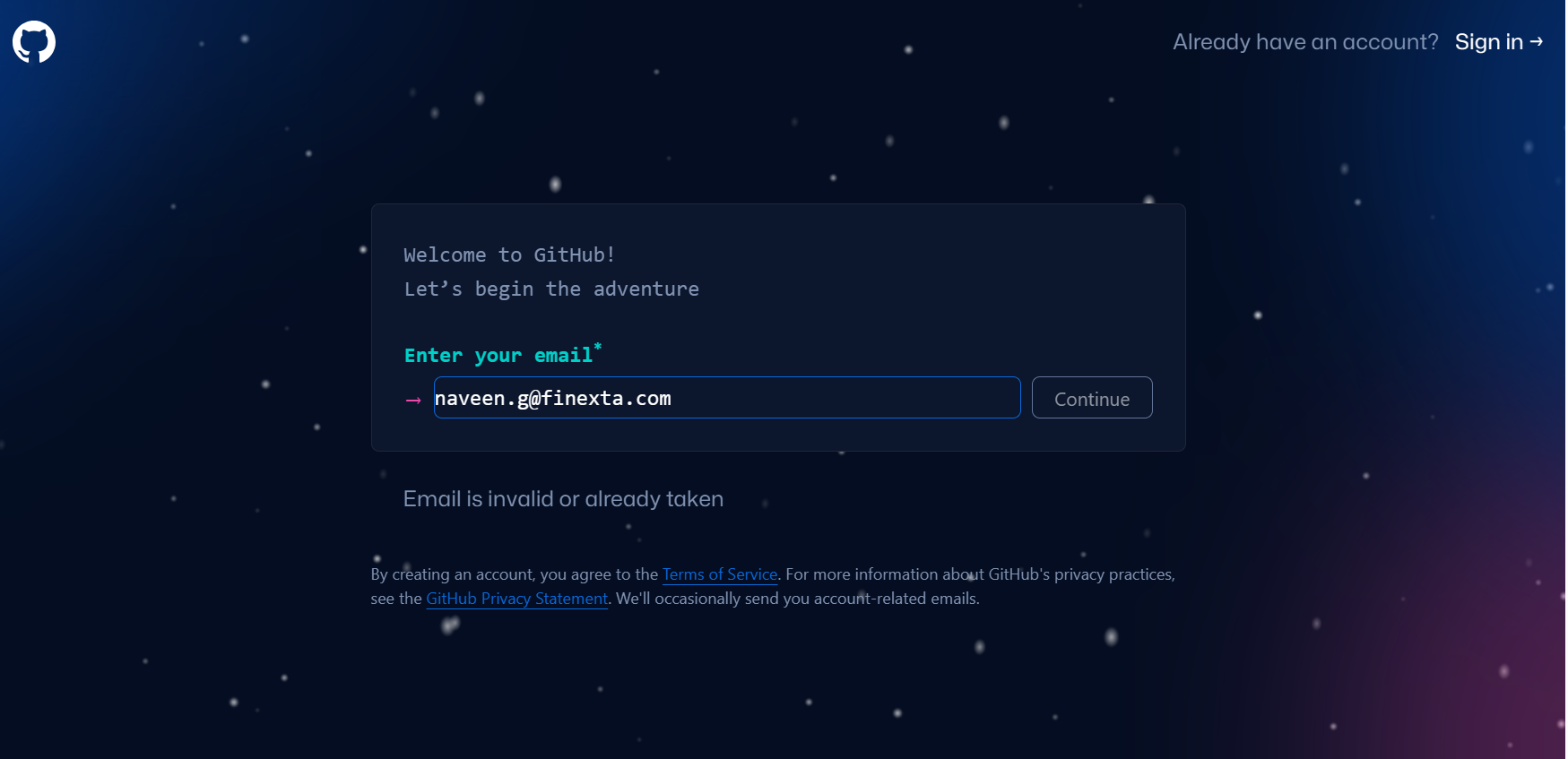


**Git** is a **version control system**, while **GitHub** is a **web-based platform** for hosting Git repositories.

# **Creation of GitHub account**

Navigate to the GitHub Home page

Open our web browser and go to [GitHub](https://github.com/)



## **2.1 GitHub Sign up**

On the GitHub homepage, we’ll find the “Sign up” button in the upper-right corner. Click on it to proceed.

Provide our Information

* + **Username**: Choose a unique username for our GitHub account. If your desired username is already taken, GitHub will suggest alternatives.
  + **Email Address**: Enter your email address. This email will be used for account notifications and communication from GitHub.
  + **Password**: Create a strong password for your account. Optionally, you can choose to receive updates and offers from GitHub

Complete the Captcha

To verify that you’re not a robot, complete the CAPTCHA as prompted.

Finish the signup

After providing the required information and completing the CAPTCHA, click the “Create account” button.

Verify our Mail

GitHub will send you an email to verify our email address. Open the mail and click on the verification link provided. If you don’t see the mail in our inbox, check our spam or junk folder.

Conformation

Once we have clicked the verification link, our GitHub account will be confirmed, and we’ll be redirected to the GitHub website.

Congratulations! You’ve successfully signed up for GitHub.!!!!!

# **Git Installation for Windows**

There are also a few ways to install Git on Windows. The most official build is available for download on the Git website. Just go to <https://git-scm.com/download/win> and the download will start automatically.

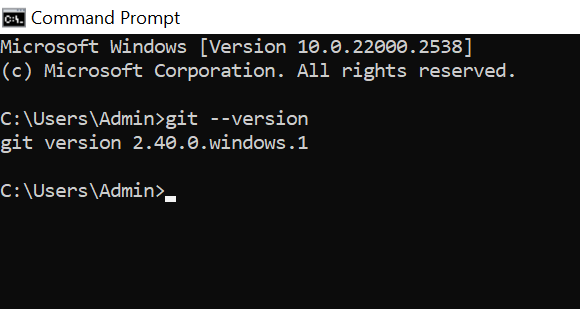
Visit the official Git website and download the Git for Windows installer.

Run the downloaded installer file. Follow the installation wizard's instructions.

we can usually accept the default settings unless you have specific preferences.

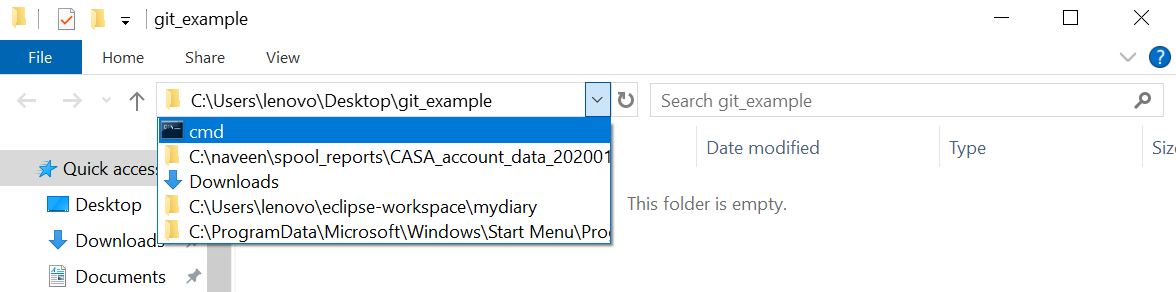
## **3.1 Verify the git installation**

Once the installation is complete, open a command prompt use git –version command it will give the git version.



# **Initialize a local git repository**

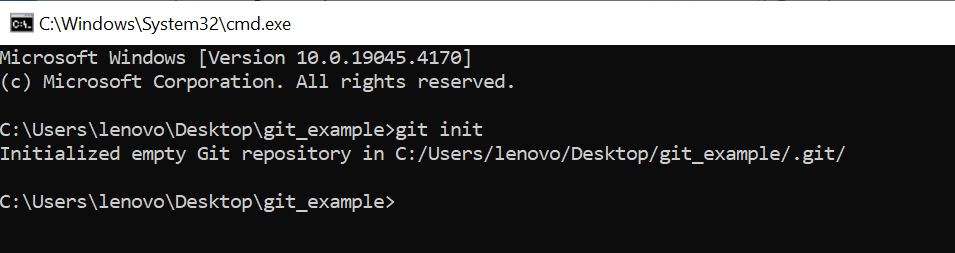
Open our project folder or create a new project folder in our system



Select the project path and give the cmd and give enter it will open the command prompt with project folder directory.

* In the command prompt, run the command *git init*to create a local Git repository

git init



# **Creation of new Repository**

**5.1 Login to GitHub**: sign in to our GitHub account

Navigate to Repository: Click the “+” icon in the top right corner and select “New repository.”

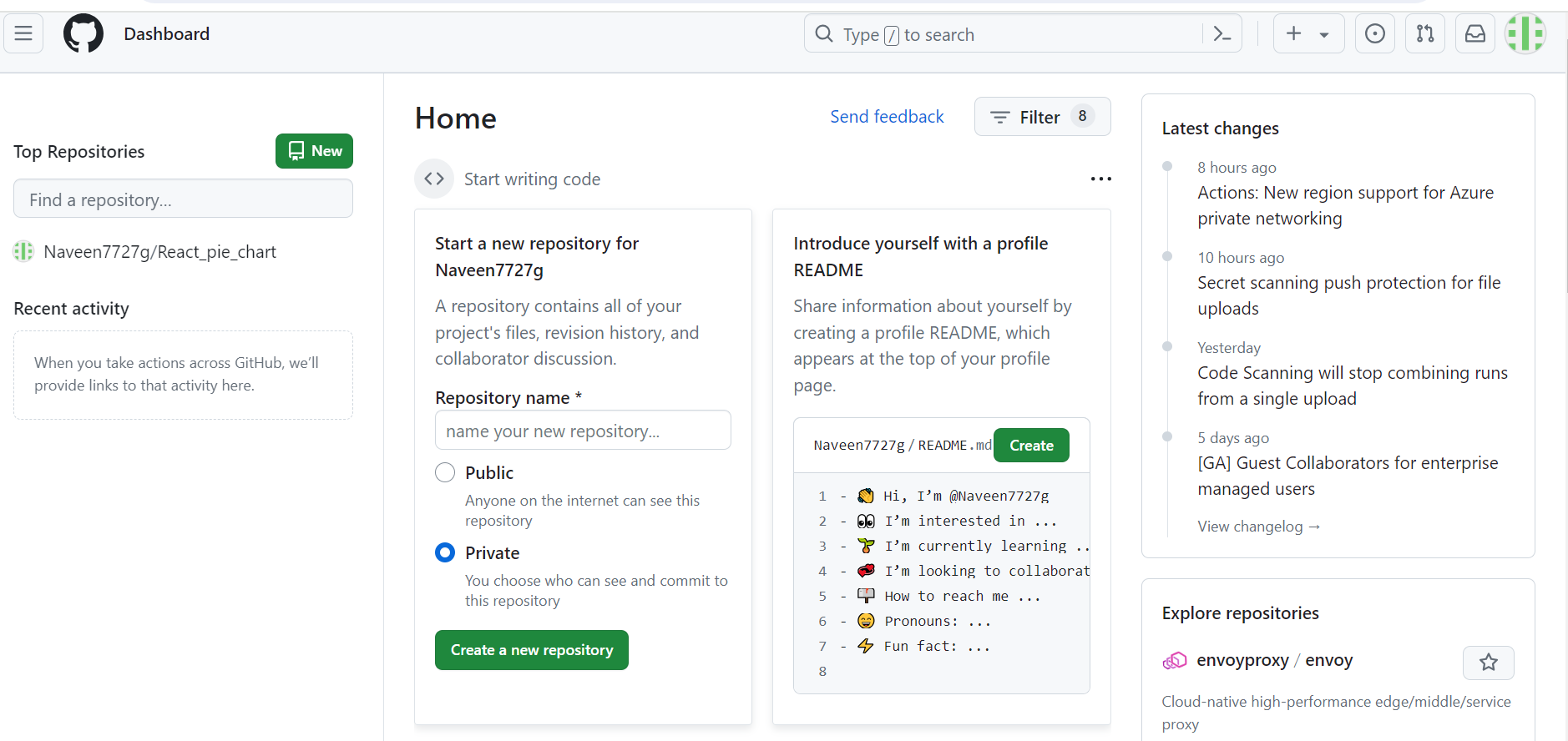
Fill in Repository Details:

* + Repository Name: Choose a memorable name.
  + Description (Optional): Add a brief description.
  + Visibility: Decide whether it should be public or private.

## **5.2 Initialize with a README**

* + Select the option to “Initialize this repository with a README.”
  + we can also create other files like .ignore and a software license.

Create Repository: Click “Create repository”



# **link our local repository to GitHub**

We should give the link our local repository to GitHub (remote repository).

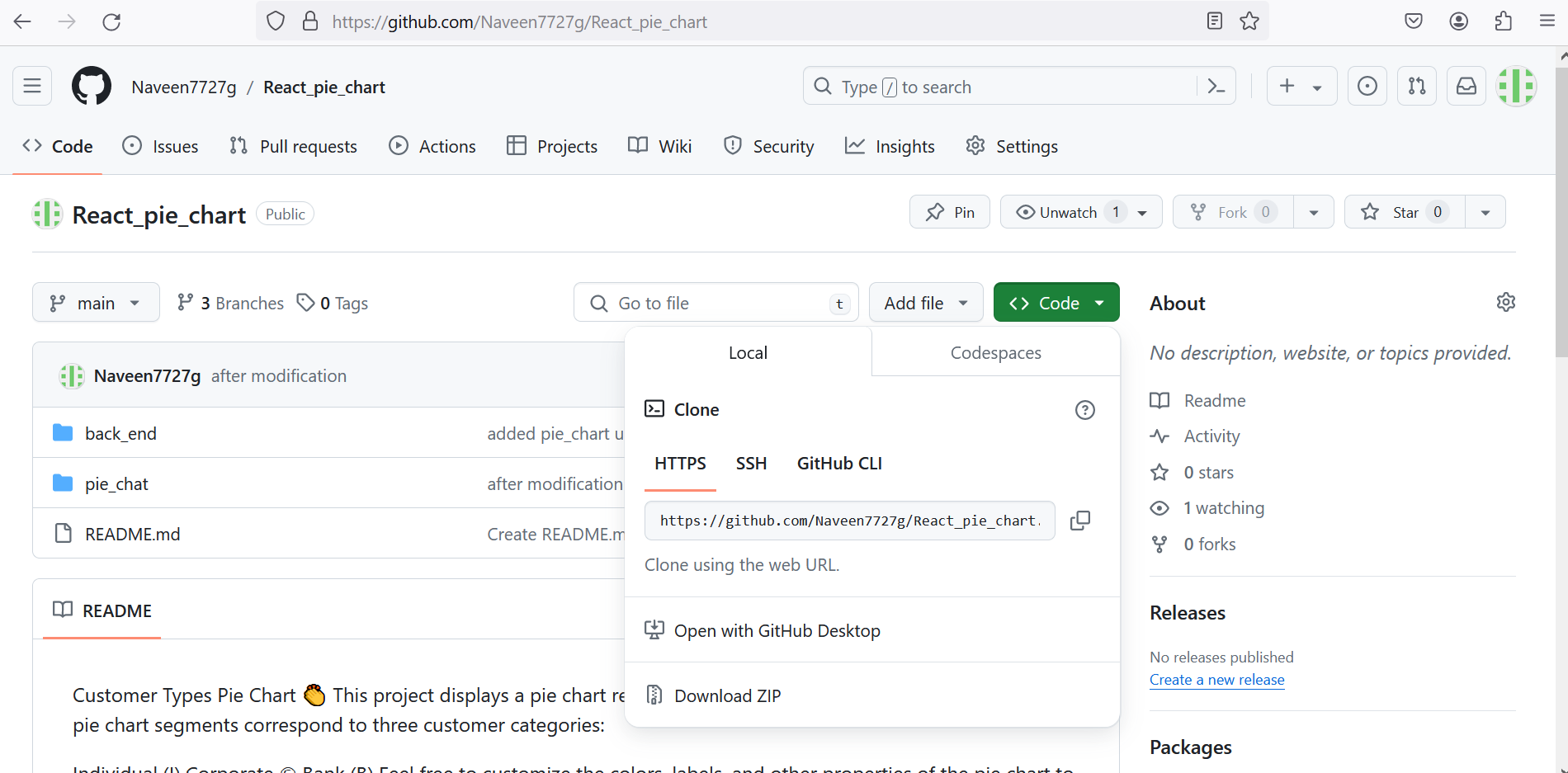
For that we have to give our GitHub repository origin URL to local repository in command prompt we should run the below command.

git remote add origin (remote repository url)

## **6.1 Get the Remote Repository URL**

First, go to the remote repository platform (such as GitHub) where our project is hosted.

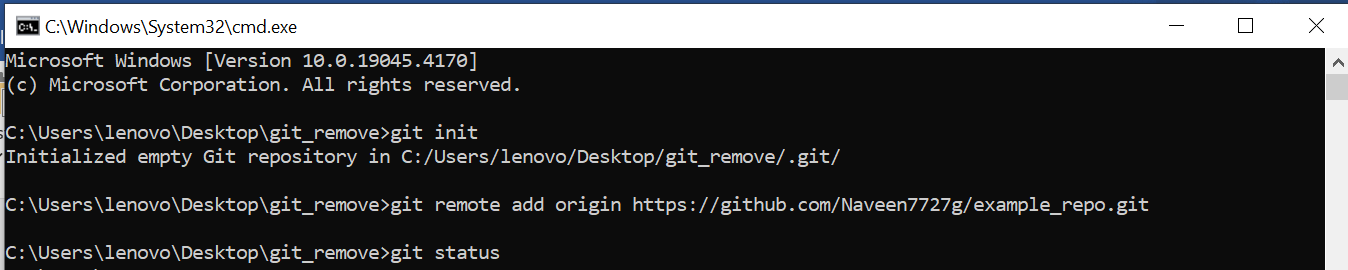
For example, see the below screen short.



* Copy the HTTPS URL or SSH URL from the top of the repository page and execute below command

Ex: git remote add origin <HTTPS\_URL>

git remote add origin <https://github.com/Naveen7727g/React_pie_chart.git>



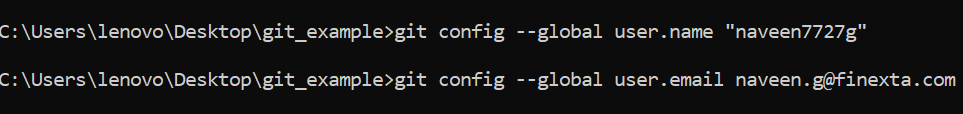
## **6.2 Configure our Git username and mail**

**Global Configuration**

Set our global Git username and mail address using the following commands

git config --global user.name "Your Name"

git config --global user.email “[youremail@yourdomain.com](mailto:youremail@yourdomain.com)”



Global configurations apply to the current user across all git repositories.

These settings are stored in ~/.gitconfig.

When we set a configuration value using above command it affects all repositories for that user.

**Local configuration**

git config --local user.name "Your Name"

git config --local user.email “[youremail@yourdomain.com](mailto:youremail@yourdomain.com)”

When we use these local configurations, it will apply to a single repository.

These settings are stored in the .git/config file within the repository.

When we set a configuration value using above command it affects only that specific repository.

We should use any one these configurations based on our requirements.

# **Push the units to Remote repository**

**Create Project Units**

We should create files in our working project directory for example

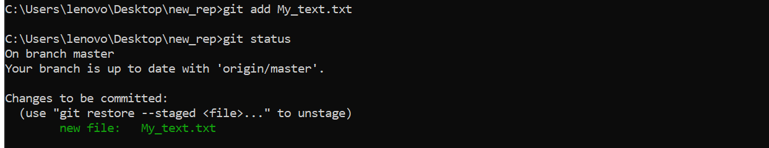
Mytext.txt

My\_file.js

Add units from working directory to staging Area

* Add our project files to the local repository using git add. Command here we can add single files also.

git add Mytext.txt



Commits the units from staging area to local repository

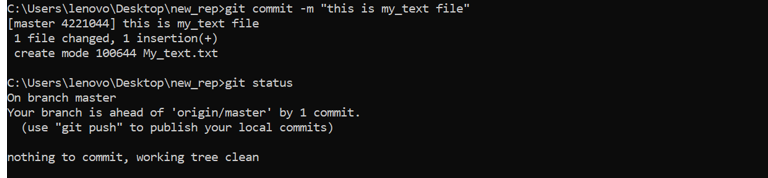
* Create an initial commit with git commit -m "Initial commit". Command here the commit command commits all files which had add to staging area.

git commit -m “message”

Above command is providing a commit message stage the changes and commit the files

* Check the status using git status command

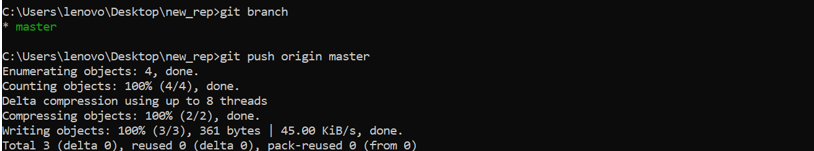
git status



Push the units from local repository to remote repository

* Push local commits to remote repository

git push -u origin main



# **Creation of branches in new repository**

**Open our Terminal or Command Prompt**

* + First, open our terminal or command prompt. we’ll be using the command line to create the branch.

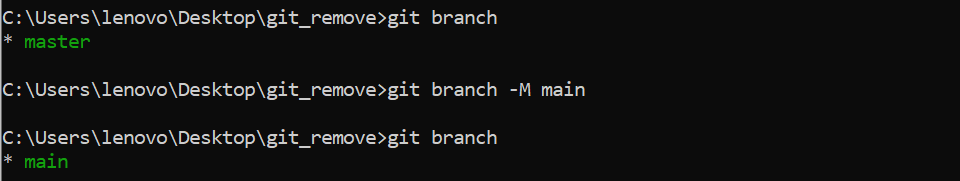
**Navigate to our Git Repository**

* + Use the cd command to navigate to the directory where our Git repository is located.

## **8.1 Modifying name of Master to required one in new repository**

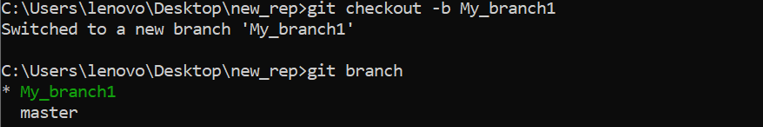
If we want to change the default branch name from “master” to “main” in our Git repository, following the below command

git branch -M main(user defined)



To create a new branch, use the following command

git checkout -b <branch-name>

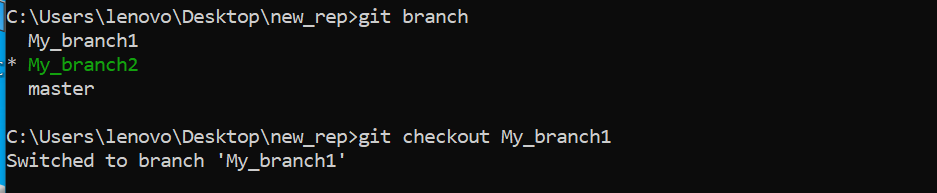


Branch list we can see by using git branch command

git branch

Switches branches we can use git checkout command

git checkout <branch name>

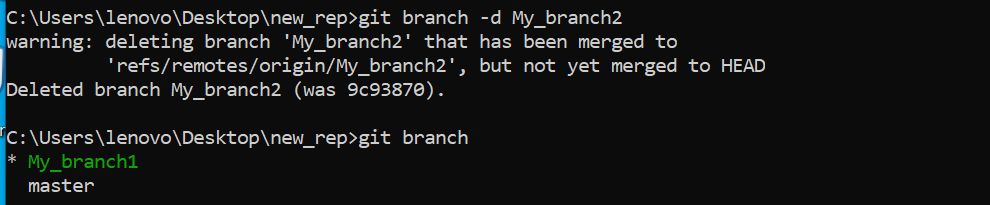


Git pull command is used to **fetch** and **integrate** changes from a **remote repository** into our **local branch.**

Git pull

Delete the branch using this command git -d branch name

git branch -d <branch\_name>

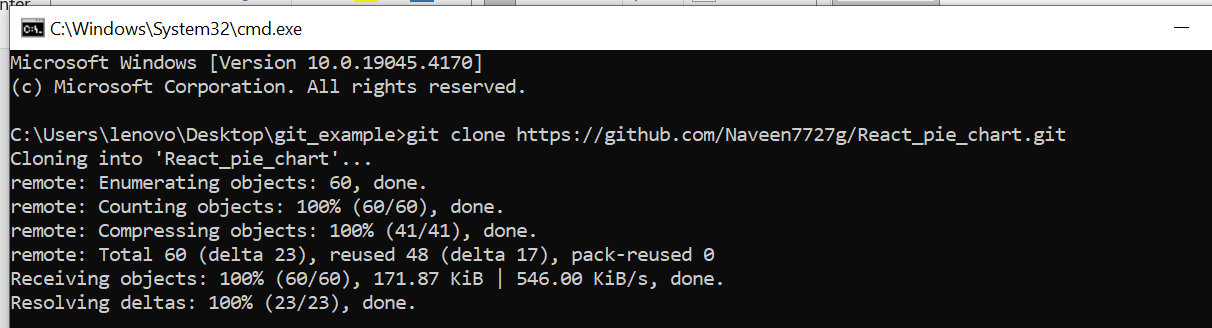


# **Clone a remote git Repository to our local Machine**

To **clone a remote Git repository to your local machine**, follow these steps:

1. **Navigate to the Repository on GitHub:**
   * Open our web browser and go to the main page of the repository we want to clone.
2. **Get the Repository URL:**
   * Above the list of files, click on the “Code” button.
   * Copy the URL for the repository (we can choose either HTTPS or SSH).
3. **Open Your Terminal or Command Prompt:**
   * Change the current working directory to the location where we want to create the cloned directory.
4. **Run the Git Clone Command:**
   * Type the following command, replacing <repository-url> with the URL we copied earlier

git clone https://github.com/Naveen7727g/React\_pie\_chart.git



# **Modifying units from existing repository**

We can create new required files in our project folder

After creation of new files can use the git add command it will add the file to our staging area.

we can use git commit command it will commits the files to local repository.

Finally, we can use git push command it will push the files to our remote repository.

git add filename

git commit -m “message for committing the files”

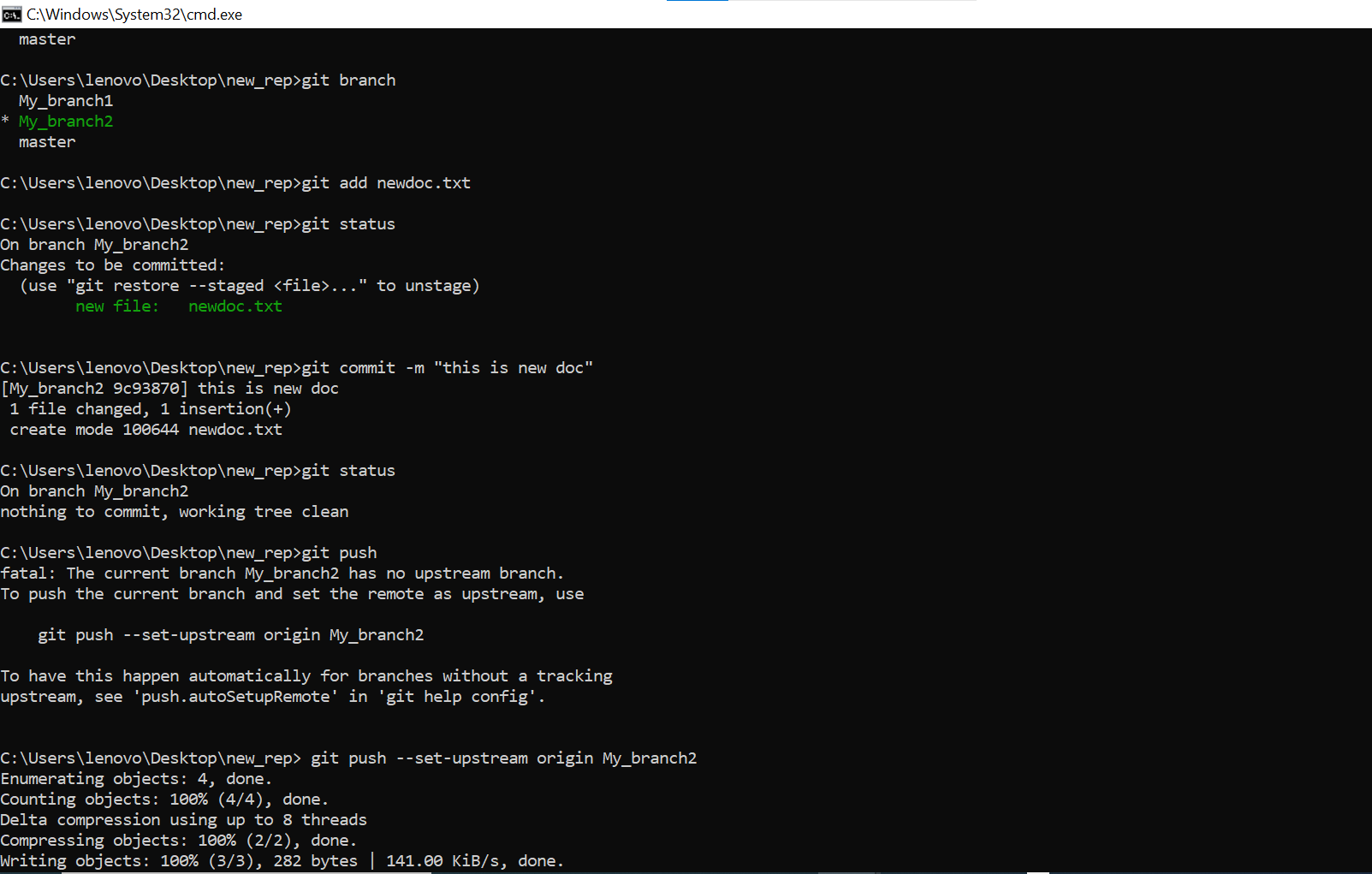
git push origin branch name

Or we can edit our existing files also, after edit we can use the same process to push edit files to our remote repository.

git add filename

git commit -m “message for committing the files”

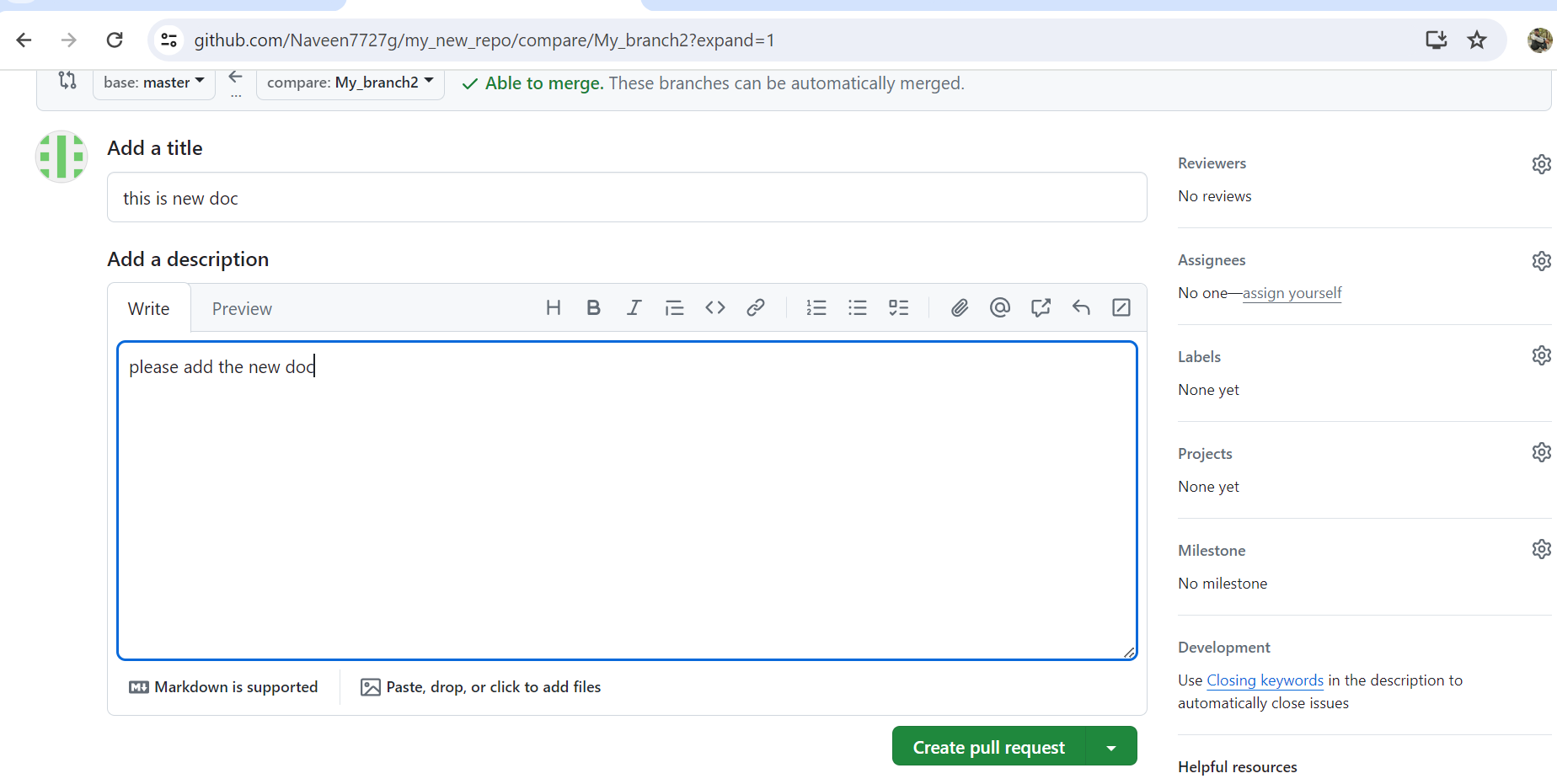
git push origin branch name



# **Merging of branches**

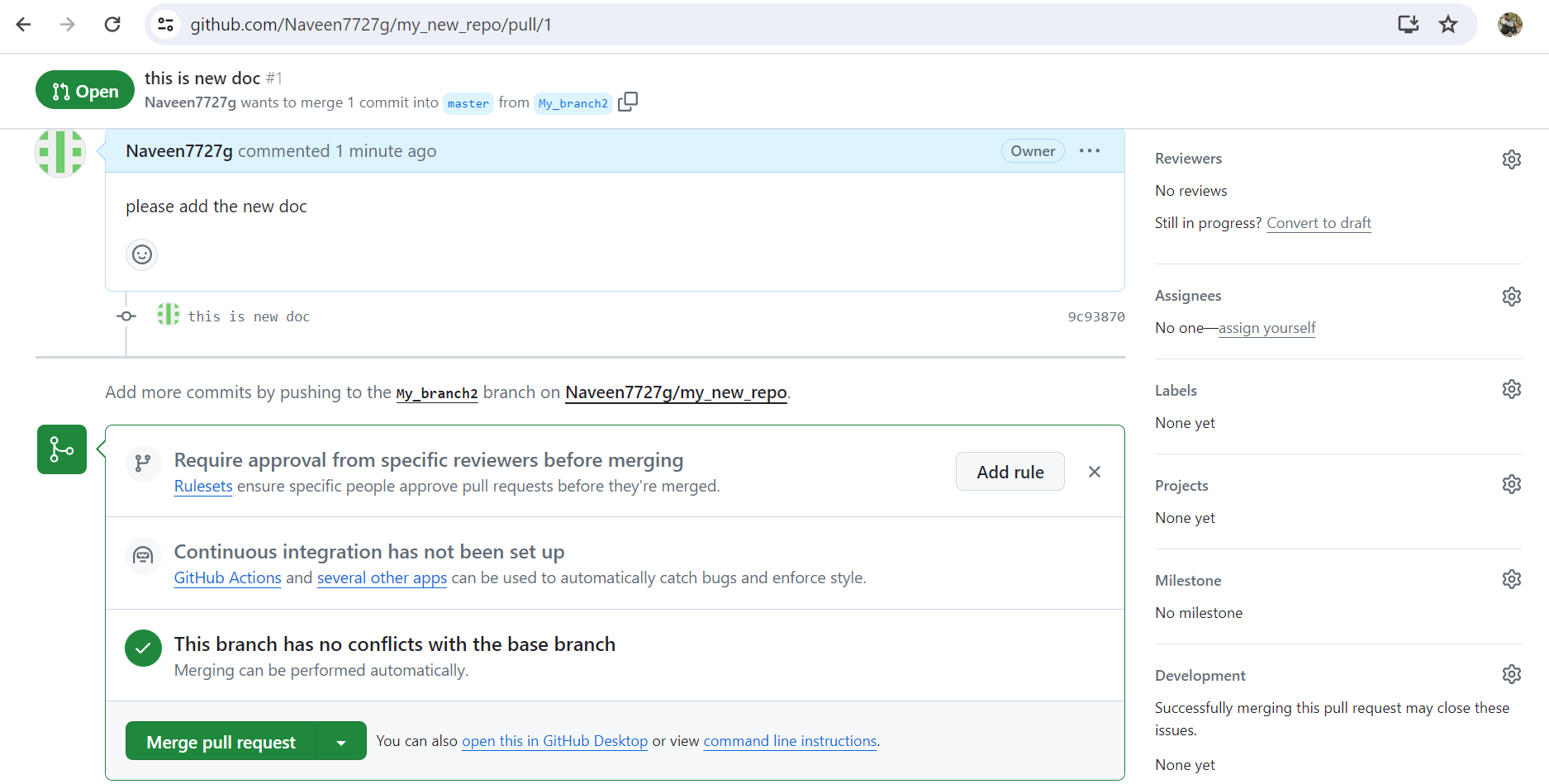
After git push in command prompt the pull request will go to our remote repository in GitHub.

When we click compare & pull request it will ask the create pull request with add title and add description.



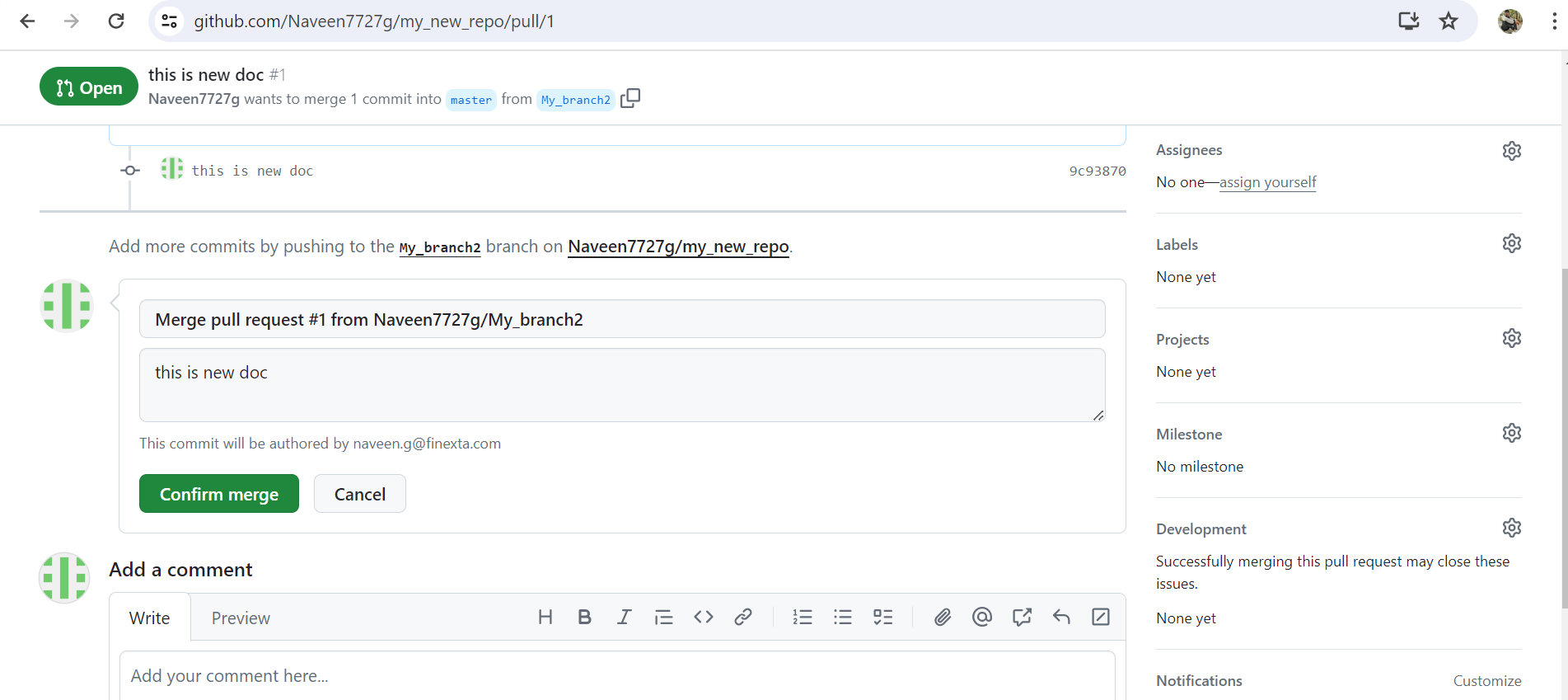
After create pull request the owner of the remote repository will merge the pull request

We can see the below screen short.



When we merge the pull request is conformed the main repository will update with all changes what ever done in the sub branch.

After git push in GitHub 🡪 compare & pull request 🡪create pull request🡪marge pull request 🡪 conform merge.



# **Merge Conflict**

A Merge conflict occurs in Git when competing changes are made to the same line of a file or when one person edits a file or while another person deletes the same file. Here are the scenarios in which merge conflicts can arise:

1. **Simultaneous Edits:**
   * When two developers modify the same lines of a file in different branches, Git cannot automatically determine which changes to keep.
   * For example, if Developer A and Developer B both edit the same function in separate branches, merging those branches may result in a conflict.
2. **File Deletion and Modification:**
   * If one developer deletes a file while another developer modifies it in a different branch, Git faces a conflict.
   * Git cannot decide whether to keep the modified content or delete the file entirely.
3. **Pending Local Changes:**
   * A merge can fail to start if there are pending changes in our working directory or staging area.
   * Git wants to avoid overwriting these local changes during the merge process.

# **Resolving a merge conflict in git involves a few steps**

1. **Identify the Conflict:**
   * Merge conflicts occur when competing changes are made to the same line of a file or when one person edits a file and another person deletes the same file.
   * We’ll see conflict markers in the file, such as <<<<<<<, =======, and >>>>>>>.
2. **Open the File:**
   * Open the conflicted file in our favorite text editor (e.g., Visual Studio Code).
3. **Review the Conflict:**
   * Look for the conflict marker <<<<<<< HEAD. This represents the changes from the current branch (HEAD).
   * Next, we’ll see =======, which separates our changes from the other branch’s changes.
   * Finally, we’ll find >>>>>>> BRANCH-NAME, representing the changes from the other branch.
4. **Resolve the Conflict:**
   * Decide which changes to keep:
     + Keep only your branch’s changes.
     + Keep only the other branch’s changes.
     + Create a new change that incorporates both branches’ changes.
   * Delete the conflict markers and make the necessary adjustments.
5. **Stage our Changes:**
   * After resolving the conflict, stage the modified file using git add.
6. **Commit the Changes:**
   * Create a new commit with the resolved conflict using git commit.
   * Provide a meaningful commit message describing the resolution.
7. **Push or Merge:**
   * If we are working with pull requests, push our changes to the remote repository and merge the branches.
   * If we are not using pull requests, we can merge the branches directly using

 git merge.

