SDM COLLEGE OF ENGINEERING AND TECHNOLOGY

Dhavalagiri, Dharwad-580002, Karnataka, India

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

A Report on:

GitHub- The Collaborative Code Hub

COURSE CODE: **22UCSE421**

COURSE TITLE: **Project Management Tools** SEMESTER: **IV**

COURSE TEACHER: **Dr.S.B.Kulkarni**

[Academic Year- 2024-25]

Submitted By:

ISHA U PRABHU	USN: 2SD23CS038
ISHANK KUMAR	USN: 2SD23CS039
JAYALAKSHMI	USN: 2SD23CS040
KARAN KUMBAR	USN: 2SD23CS041

Table of Contents

1.ABOUT GITHUB

GitHub is a **web-based platform** which hosts software development projects and uses Git for version **management**. **Git** is a **distributed version control system** that helps developers to work together on same software projects and **keep track of changes** made to their code by on another. GitHub offers a user-friendly interface, which is very **collaborative** tools, and more **project management** tools, GitHub will enhance the potential of the Git.

GitHub allows developers to create and **manage the code in the repository** in a remote location where others can access the code or GitHub is a **collection repository** that contains the files **of the project**. To understand exactly what GitHub is, you need to know **two connected principles**:

- **Version control**
- **Git**

Version control, also known as source control, is the practice of **tracking** and **managing changes** to software code. Version control systems are software tools that help software teams manage changes to source code over time. As development environments have accelerated, version control systems help **software teams work faster and smarter.** They are especially useful for DevOps teams since they help them to reduce development time and increase successful deployments.

Version control software keeps track of every modification to the code in a special kind of database. **If a mistake is made, developers can turn back the clock and compare earlier versions of the code to help fix the mistake while minimizing disruption to all team members.**

By far, the most widely used modern version control system in the world today is **Git.** Git is a mature, actively maintained **open-source project** originally developed in **2005 by Linus Torvalds**, the famous creator of the Linux operating system kernel.

A staggering number of software projects rely on Git for version control, including commercial projects as well as open source. Developers who have worked with Git are well represented in the pool of available software development talent and it works well on a wide range of operating systems and IDEs (Integrated Development Environments).

Having a distributed architecture, Git is an example of a DVCS (hence **Distributed Version Control System**). Rather than have only one single place for the full version history of the software as is common in once-popular version control systems like CVS or Subversion, in Git, every developer's working copy of the code is also a repository that can contain the full history of all changes.

In addition to being distributed, **Git has been designed with performance, security and flexibility** in mind.

How do Git and GitHub work together?

When you upload files to GitHub, you'll store them in a "**Git repository**." This means that when you make changes (or "commits") to your files in GitHub, Git will automatically start to **track and manage your changes**.

There are plenty of Git-related actions that you can complete on GitHub directly in your browser, such as creating **a Git repository**, creating **branches**, and **uploading and editing files.**

However, most people work on their files **locally** (on their own computer), then continually sync these local changes—and all the related Git data—with the central "**remote"** repository on GitHub. There are plenty of tools that you can use to do this, such as **GitHub Desktop.**

Once you start to collaborate with others and all need to work on the same repository at the same time, you'll continually:

Pull all the latest changes made by your collaborators from the **remote repositor**y on GitHub.

Push back your own changes to the **same remote repository** on GitHub.

Git figures out how to **intelligently merge** this flow of changes, and GitHub helps you manage the **flow through features** such as **"pull requests."**

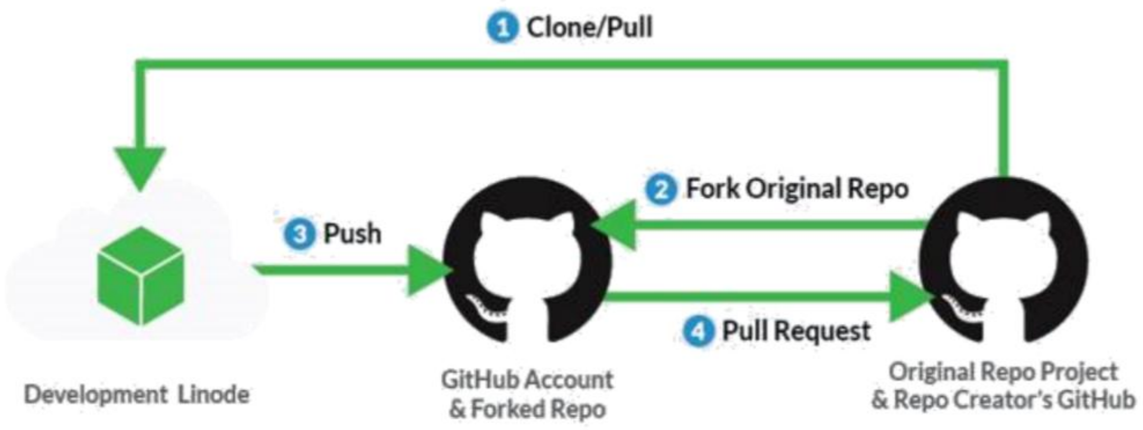

1 Clone/Pull
2 Fork Original Repo
3 Push
4 Pull Request
Development Linode
GitHub Account & Forked Repo
Original Repo Project & Repo Creator's GitHub

Fig: Git and GitHub

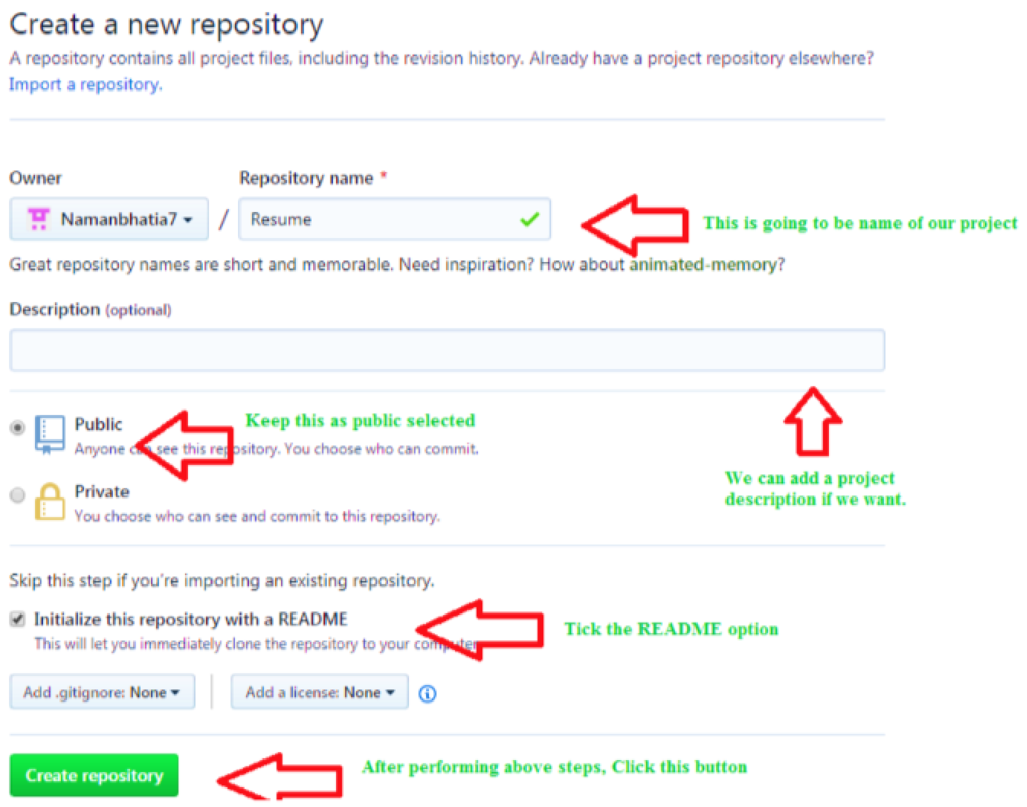
2. GITHUB FEATURES

GitHub has the following features:

1. Create and Update Repositories
2. Add Files to Your Repository
3. Operations in GitHub
3.1 Branches
3.2 Cloning Repositories
3.3 Commit changes
3.4 Create a Pull request
3.5 Merging a Pull request

Feature 1: Create and Update Repositories.

The most important use case of **GitHub** is its ability as a **hosting platform to store files** on its website. GitHub provides developers with the **tools to Create Repositories, update repositories** either by editing files, or adding/deleting files.


Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.

Owner: Namanbhatia7 / Repository name *: Resume

This is going to be name of our project

Great repository names are short and memorable. Need inspiration? How about animated-memory?

Description (optional)

We can add a project description if we want.

Public — Anyone can see this repository. You choose who can commit.

Keep this as public selected

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

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

Initialize this repository with a README — This will let you immediately clone the repository to your computer.

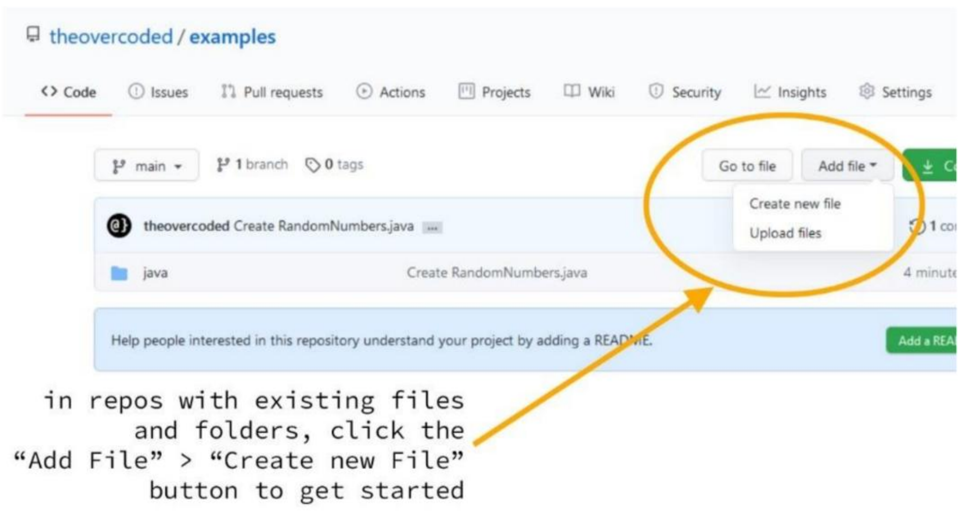
Tick the README option

Add .gitignore: None | Add a license: None

After performing above steps, Click this button

Feature 2: Add Files to Your Repository.

Again, there are three ways to add files to your repository. One is using **Git Bash**, then with **GitHub** webpage UI and the other uses the **Git GUI**.



Help people interested in this repository understand your project by adding a README.

in repos with existing files and folders, click the "Add File" > "Create new File" button to get started

Feature 3: Operations in GitHub

These are common procedures that have been performed on files in GitHub repositories with varying use cases. These are used to **provide platform users** with the ability to change the contents of their repositories dynamically.

Some of the common operations in GitHub are:

3.1 Creating a branch.

3.2 Cloning a repository.

3.3 Commit changes

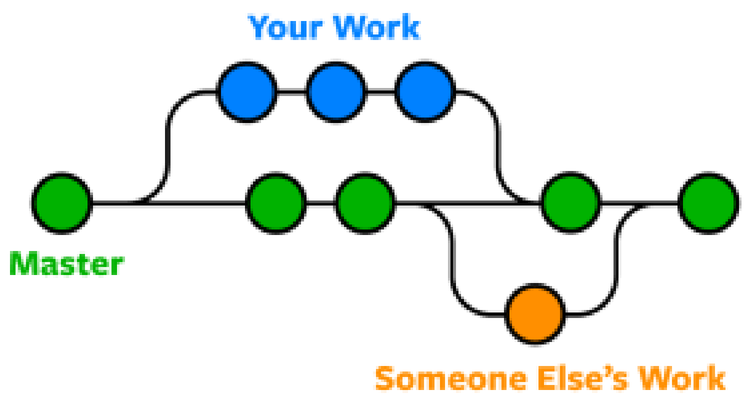
3.4 Creating pull request.

3.5 Merging pull requests.

3.6 Pushing into GitHub repository.

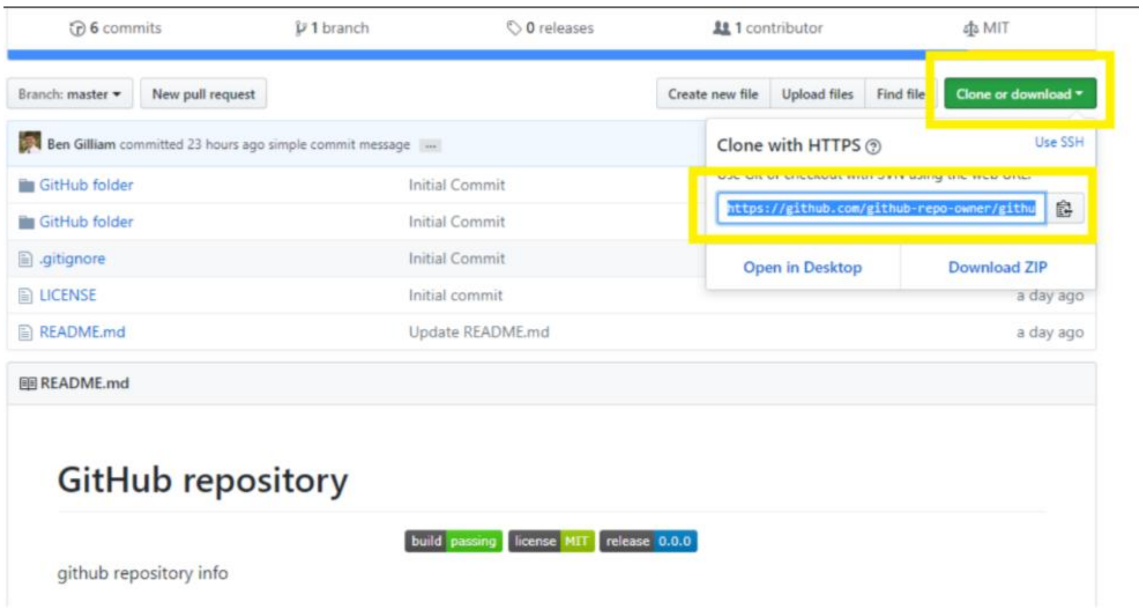
3.1 Branches.

The branches in GitHub allow you to **experiment safely** with your code**.** Branches provide a safe space for **code experimentation** and **debugging processes**. Branches always start from the **"main"** repository and branch out to **sub-repositories** to test specific parts of the code in different environments.



3.2 Cloning Repositories.

GitHub offers many features such as cloning repositories. This is done **to keep a copy of the repository on your desktop**, and the **changes you make in the Desktop files** will also be **reflected** on the GitHub page.



6 commits | 1 branch | 0 releases | 1 contributor | MIT

Branch: master | New pull request | Create new file | Upload files | Find file | Clone or download

Ben Gilliam committed 23 hours ago simple commit message

GitHub folder	Initial Commit	
GitHub folder	Initial Commit	
.gitignore	Initial Commit	
LICENSE	Initial commit	a day ago
README.md	Update README.md	a day ago

Clone with HTTPS | Use SSH

Use Git or checkout with SVN using the web URL.

https://github.com/github-repo-owner/githu

Open in Desktop | Download ZIP

README.md

GitHub repository

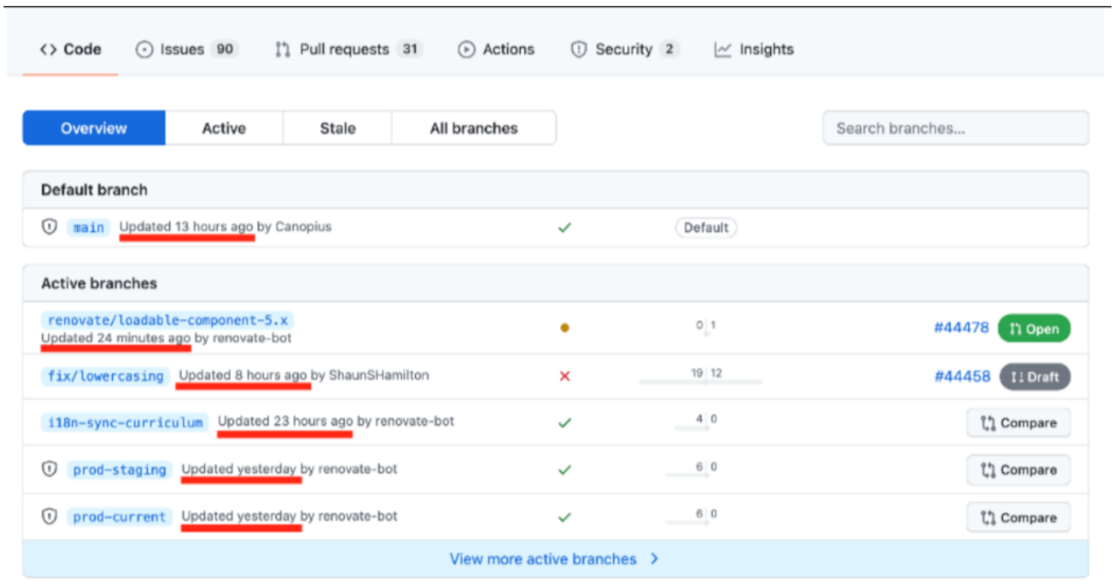
build passing | license MIT | release 0.0.0

github repository info

3.3 Commit Changes

Similar to saving a file that's been edited, a commit **records changes** to one or more files in your branch. Git assigns each commit a **unique ID, called a SHA or hash,** that identifies:

- **The specific changes.**
- **When the changes were made.**
- **Who created the changes.**



<> Code | Issues 90 | Pull requests 31 | Actions | Security 2 | Insights

Overview | Active | Stale | All branches | Search branches...

Default branch

main	Updated 13 hours ago by Canopius	✓	Default	

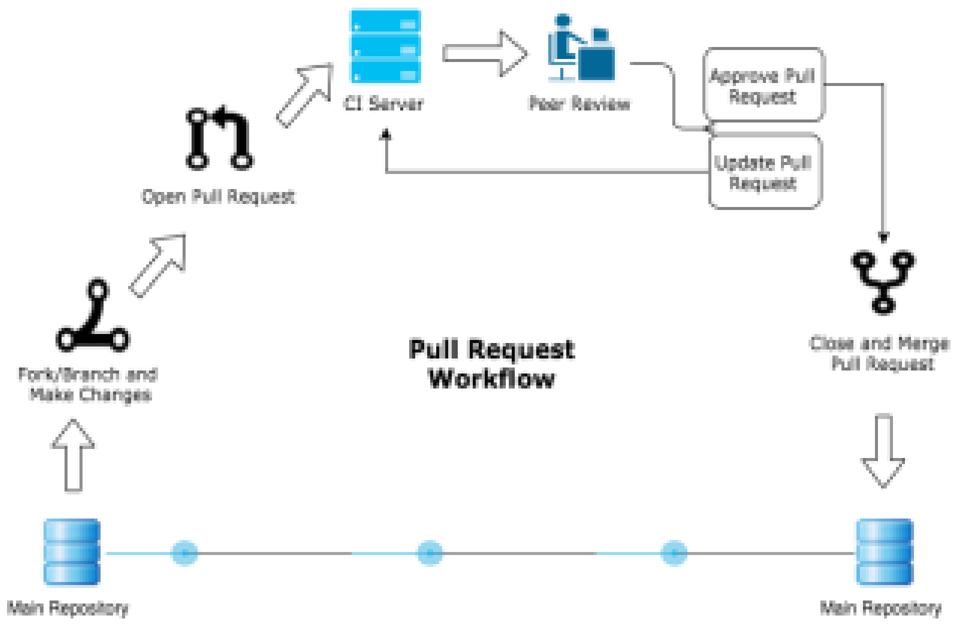
Active branches

renovate/loadable-component-5.x	Updated 24 minutes ago by renovate-bot	●	0 | 1	#44478	Open
fix/lowercasing	Updated 8 hours ago by ShaunSHamilton	×	19 | 12	#44458	Draft
i18n-sync-curriculum	Updated 23 hours ago by renovate-bot	✓	4 | 0		Compare
prod-staging	Updated yesterday by renovate-bot	✓	6 | 0		Compare
prod-current	Updated yesterday by renovate-bot	✓	6 | 0		Compare

View more active branches >

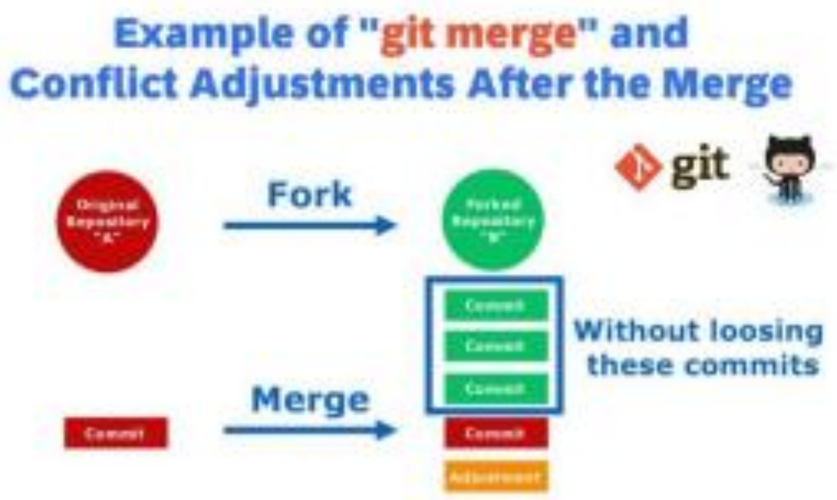
3.4 Creating a Pull request

Create a pull request to **propose and collaborate** on **changes** to a repository. These changes are proposed in a **branch,** which ensures that the **default branch** only contains **finished and approved work**.'



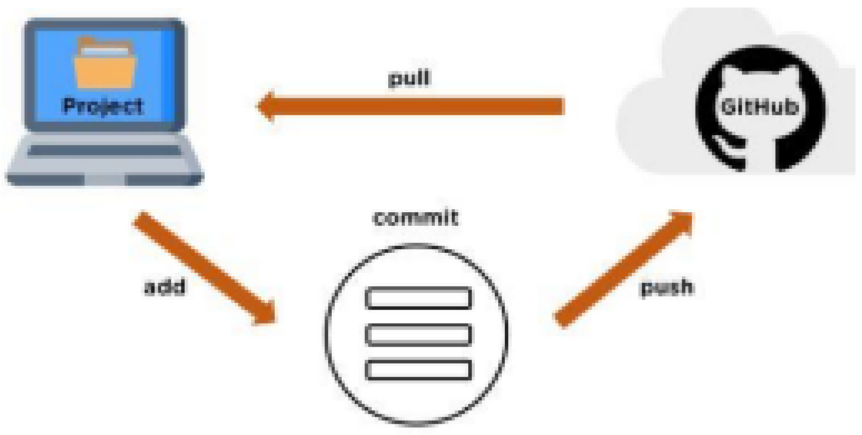
3.5 Merging a Pull request.

In a pull request, you propose that **changes you've made on a feature branch** should be merged into a **base branch** **(main branch)**. By default, any pull request can be merged at any time, unless the head branch is in conflict with the base branch. However, there may be restrictions on when you can merge a pull request into a specific branch.



3.6 Pushing into GitHub repository.

git push **updates the remote branch with local commits**. It is one of the four commands in Git that **prompts interaction** with the remote repository. You can also think of git push as **update *or* publish**.
By default, git push only updates the corresponding branch on the remote. So, if you are checked out to the main branch when you execute git push, then only the main branch will be updated. It's always a good idea to use git status to see what branch you are on before pushing to the remote.



3.7 CHART SHOWING ALL FEATURES OF GITHUB

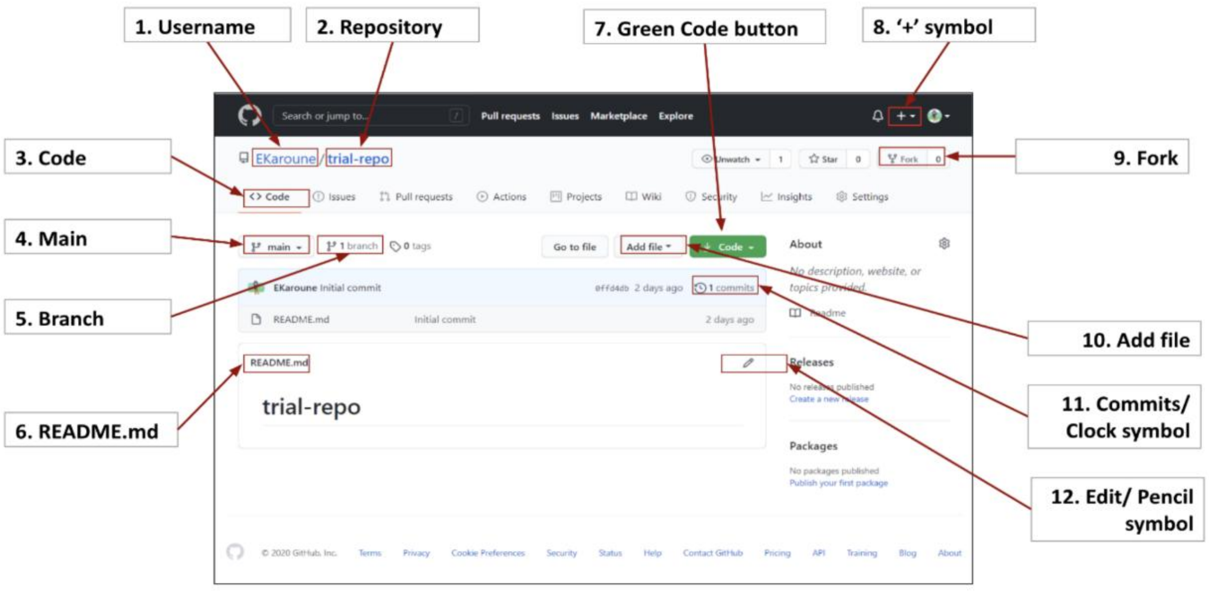


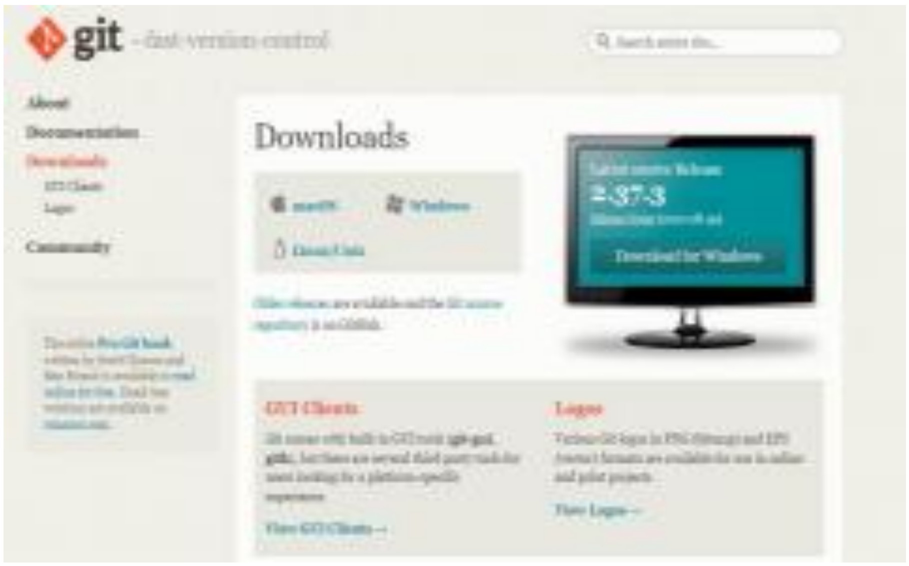
Figure: chart showing all features of GitHub.

<u>3.INSTALLATION PROCEDURE</u>

<u>Step 1: Install Git-Bash</u>

1.Browse to the official Git website: https://git-scm.com/downloads .

2.Click the download link for Windows and allow the download to complete.



3. Go to the download location (or use your browser's shortcut) and double-click the file to launch the installer.
4. Click **Yes** on the User Account Control prompt.
5. Review the license, then click **Next**.
6. Choose (or keep) the default installation location and click **Next**.
7. On the component screen, leave defaults and click **Next**.
8. Click **Next** to create a Start menu folder.
9. Choose your preferred text editor (e.g., Notepad++) and click **Next**.
10. Keep the default branch name ('master') unless required otherwise and click **Next**.
11. Leave the recommended PATH option selected and click **Next**.
12. Use the default SSH client and click **Next**.

13. Leave the server certificate option as default (change only if in Active Directory), then click **Next**.
14. Keep the default line ending conversion setting and click **Next**.
15. Use the default terminal emulator (MinTTY) and click **Next**.
16. Keep the default git pull behaviour and click **Next**.
17. Leave the default credential helper and click **Next**.
18. Enable symbolic links only if needed, then click **Next**.
19. Leave experimental features unchecked unless needed, then click **Install**.
20. Once done, tick boxes as needed (e.g., Launch Git Bash) and click **Finish**. To open Git Bash later, search "git bash" in the Start menu.

Step 2: Download VISUAL STUDIO CODE

Step 1: Visit the https://code.visualstudio.com/docs/?dv=win of the using any web browser like Google Chrome, Microsoft Edge.

Step 2: Press the "Download for Windows" button on the website to start the download of the Visual Studio Code Application.

Step 3: When the download finishes, then the **Visual Studio Code Icon** appears in the downloads folder.

Step 4: Click on the **Installer** icon to start the installation process of the Visual Studio Code.

Step 5: After the Installer opens, it will ask you to accept the terms and conditions of the Visual Studio Code. Click on **I accept the agreement** and then click the **Next** button.

Step 6: Choose the location data for running the Visual Studio Code. It will then ask you to browse the location. Then click on the **Next** button.

Step 7: Then it will ask to begin the installation setup. Click on the **Install** button.

Step 8: After clicking on Install, it will take about 1 minute to install the Visual Studio Code on your device.

Step 9: After the Installation setup for Visual Studio Code is finished, it will show a window like this below. Tick the "**Launch Visual Studio Code**" checkbox and then click **Next**.

Step 10: After the previous step, the **Visual Studio Code window** opens successfully. Now you can create a new file in the Visual Studio Code window and choose a language of yours to begin your programming journey!

Step 3: Create a GitHub Account:

To create an account on GitHub, you will be asked for some personal **information like name, confirm your email, set a username and password**, and your account should be set up in minutes.

You will be asked to **authenticate** your GitHub account, so just sign in with the same email to confirm

Step 4. Connect your GitHub account to your Git account.

You'll do this from your Git Bash terminal.

To set your Git username, type this in your terminal:

```
git config --global user.name "your user name"
```

To set your Git email, type this in your terminal:

```
git config --global user. Email "your email address"
```

4.ILLUSTRATION OF FEATURES AND ENVIRONMENT

1.Create and Update Repositories

1. In the upper-right corner of any page, select, then click **new repository**.

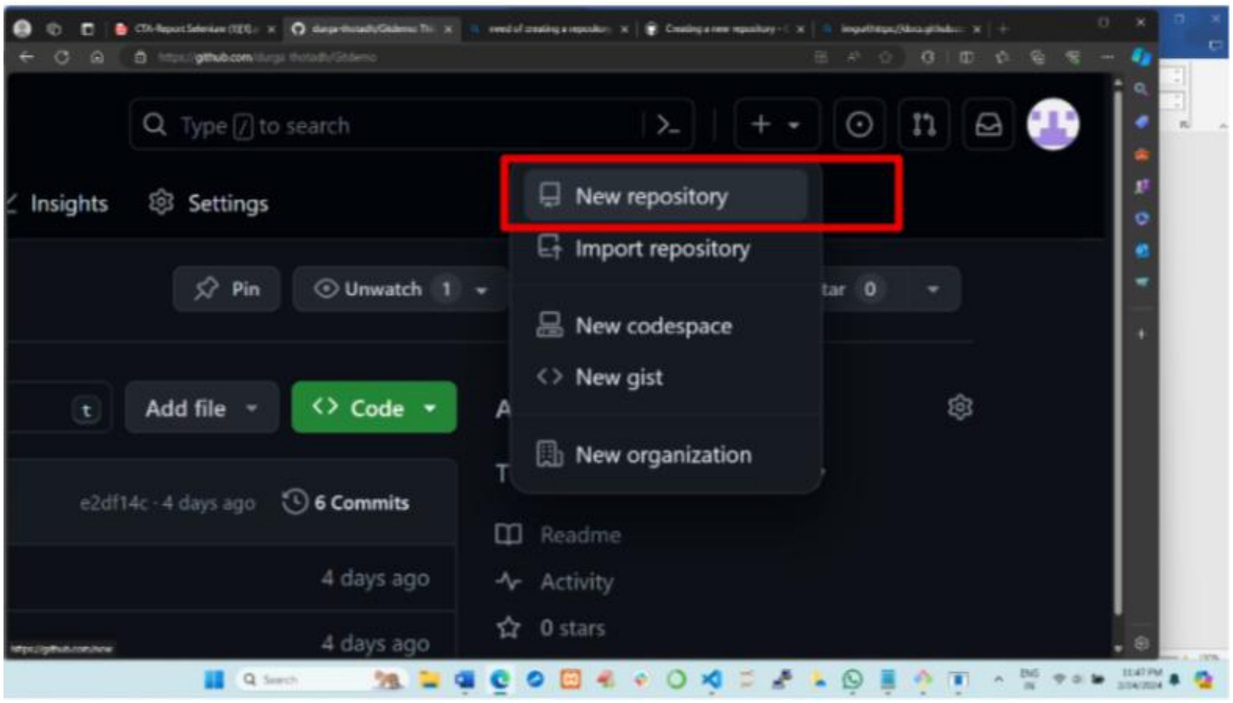


Figure: Creating a new repository

2. In the "Repository name" box, type hello-world.

3. In the "Description" box, type a short description. For example, type "This repository is for practising the GitHub Flow."
4. Select whether your repository will be **Public** or **Private**.

5. Select **Add a README file**.

6. Click **Create repository**.

2. Adding a file to a repository on GitHub

1. On GitHub.com, navigate to the main page of the repository.

2. Above the list of files, select the **Add file** dropdown menu and click **Upload files**. Alternatively, you can drag and drop files into your browser.

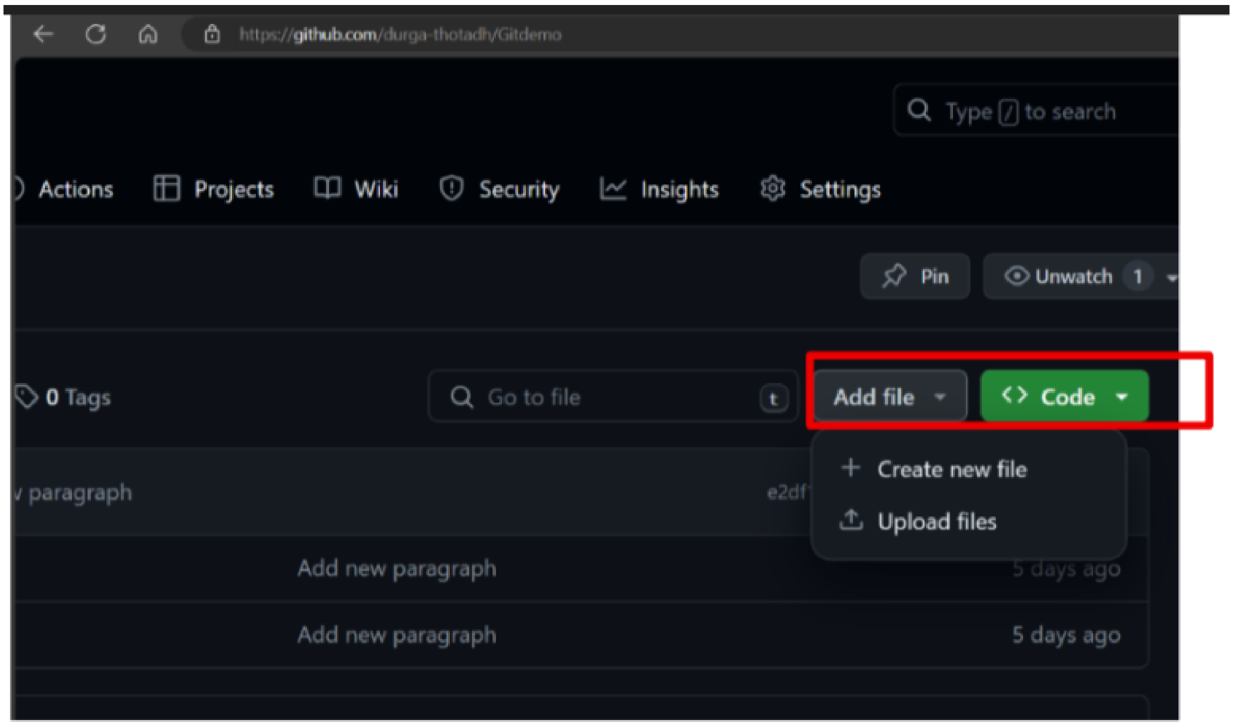


Figure: Adding a file to repository

3. To select the files you want to upload, drag and drop the file or folder, or click **choose your files**.
4. In the "Commit message" field, type a short, meaningful commit message that describes the change you made to the file

3.Operations in GitHub

3.1 Branches

1. Click the **Code** tab of your repository.

2. Above the file list, click the dropdown menu that says **main**.

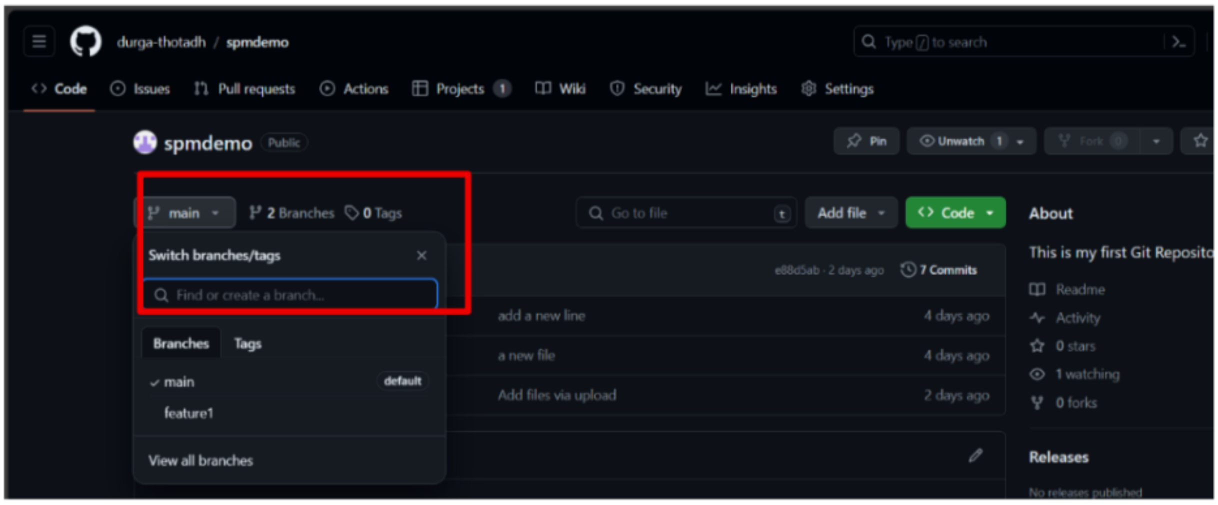


Figure: switching branches or tags

3. Type a branch name, into the text box.

4. Click **Create branch: from main**.

3.2 Cloning Repositories

You can clone your existing repository or clone another person's existing repository to contribute to a project.

1. Cloning a repository

2. On GitHub.com, navigate to the main page of the repository.
3. Above the list of files, click **Code**.

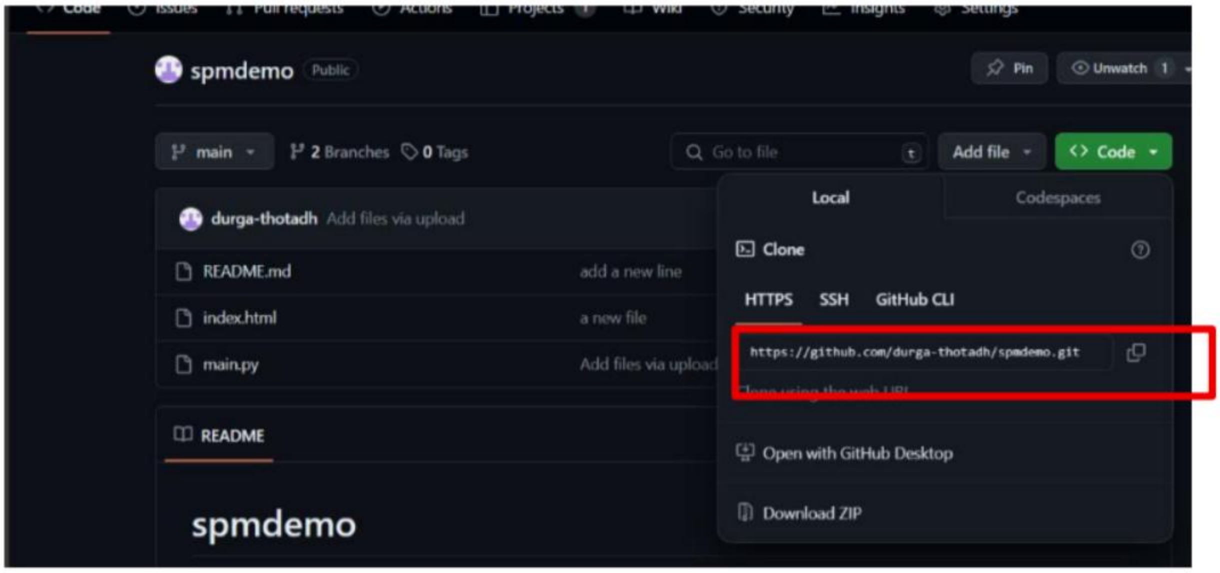


Figure: copying HTTPs URL for cloning into local repository

4. Copy the URL for the repository.

5. To clone the repository using HTTPS, under "HTTPS", click.

6. To clone the repository using an SSH key, including a certificate issued by your organization's SSH certificate authority, click **SSH**, then click.
7. To clone a repository using GitHub CLI, click **GitHub CLI**, then click.

Open Git Bash.

1. Change the current working directory to the location where you want the cloned directory. 2. Type git clone, and then paste the URL you copied earlier.
3. git clone https://github.com/YOUR-USERNAME/YOUR-REPOSITORY
4. Press **Enter** to create your local clone.

$ git clone https://github.com/YOUR-USERNAME/YOUR-REPOSITORY

3.3 Commit changes

1. Under the branch you created, click the README.md file.

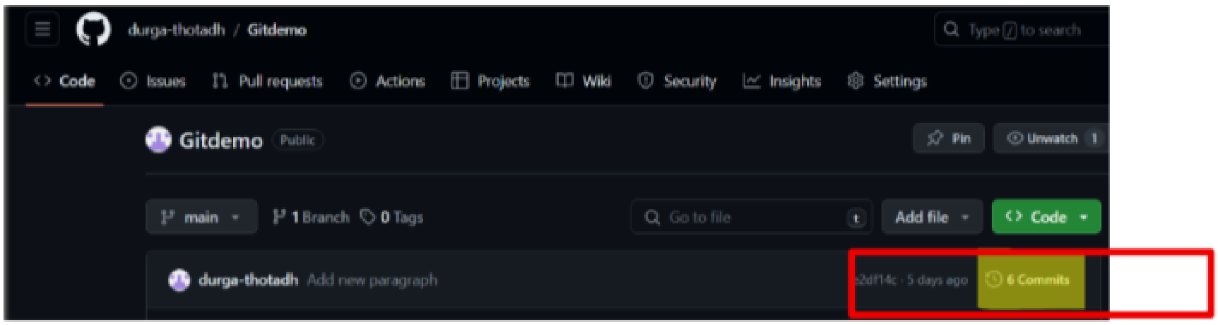
2. To edit the file, click.

3. In the editor, write a bit about yourself.

4. Click **Commit changes...**.

5. In the "Commit changes" box, write a commit message that describes your changes.

6. Click **Commit changes**.
7. These changes will be made only to the README file on your branch, so now this branch contains content that's different from main.



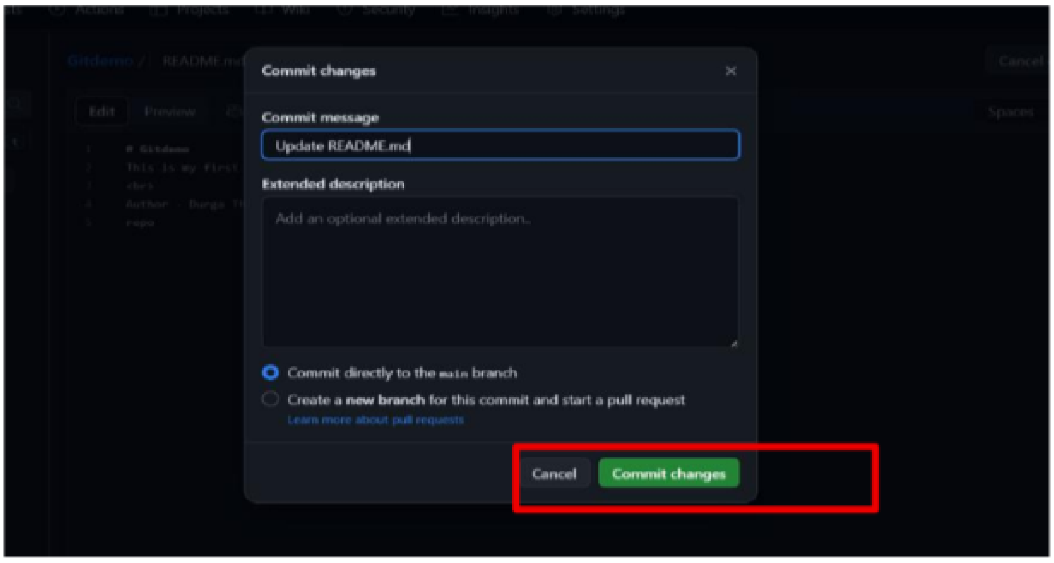

Commit changes
Commit message
Update README.md
Extended description
Add an optional extended description..
Commit directly to the main branch
Create a new branch for this commit and start a pull request
Learn more about pull requests
Cancel
Commit changes

Figure: committing changes on README.md file

3.4 Create a Pull request

1. On GitHub.com, navigate to the **main page** of the repository. 2. In the "**Branch"** menu, choose the branch that contains your commits.

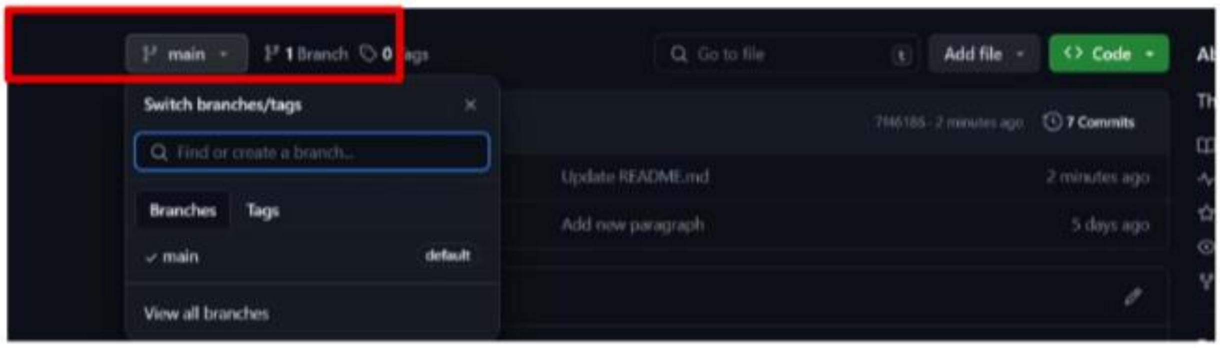


Figure: choosing a branch for pulling a request

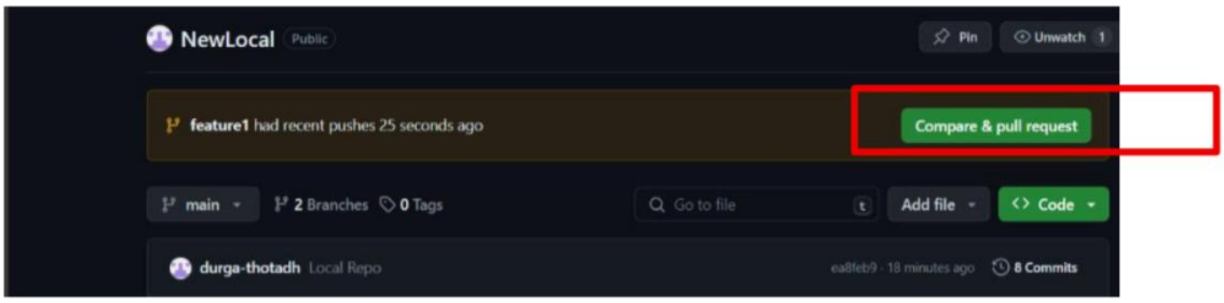


Figure: feature 1 branch is selected for compare and pull request

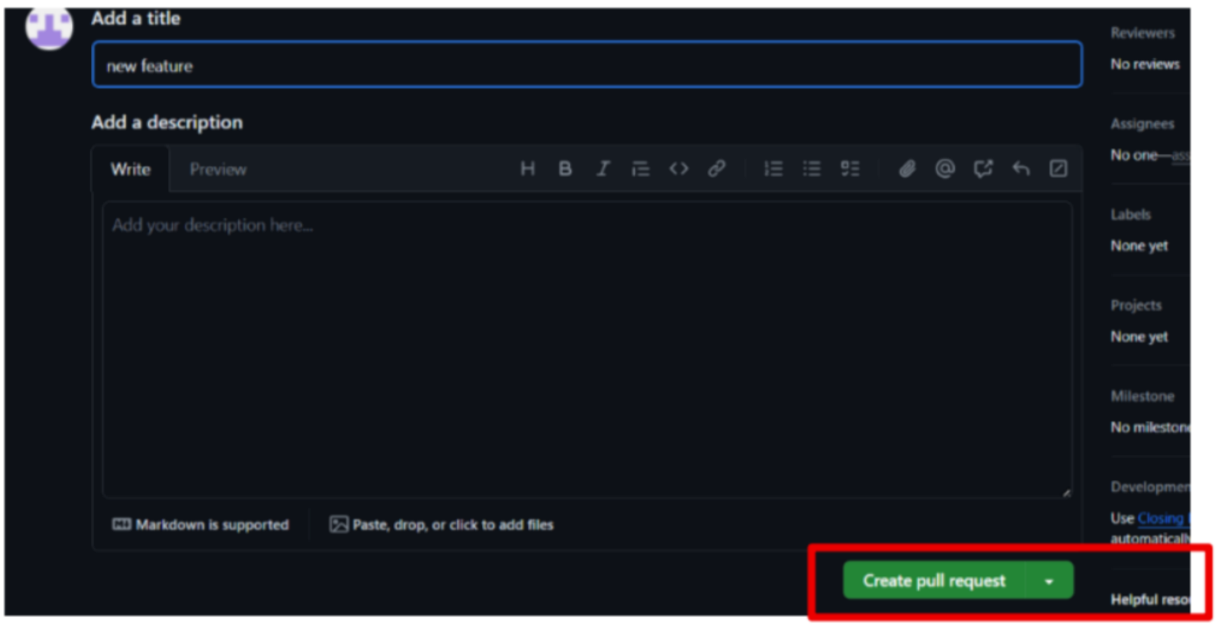


Figure: creating a new pull request

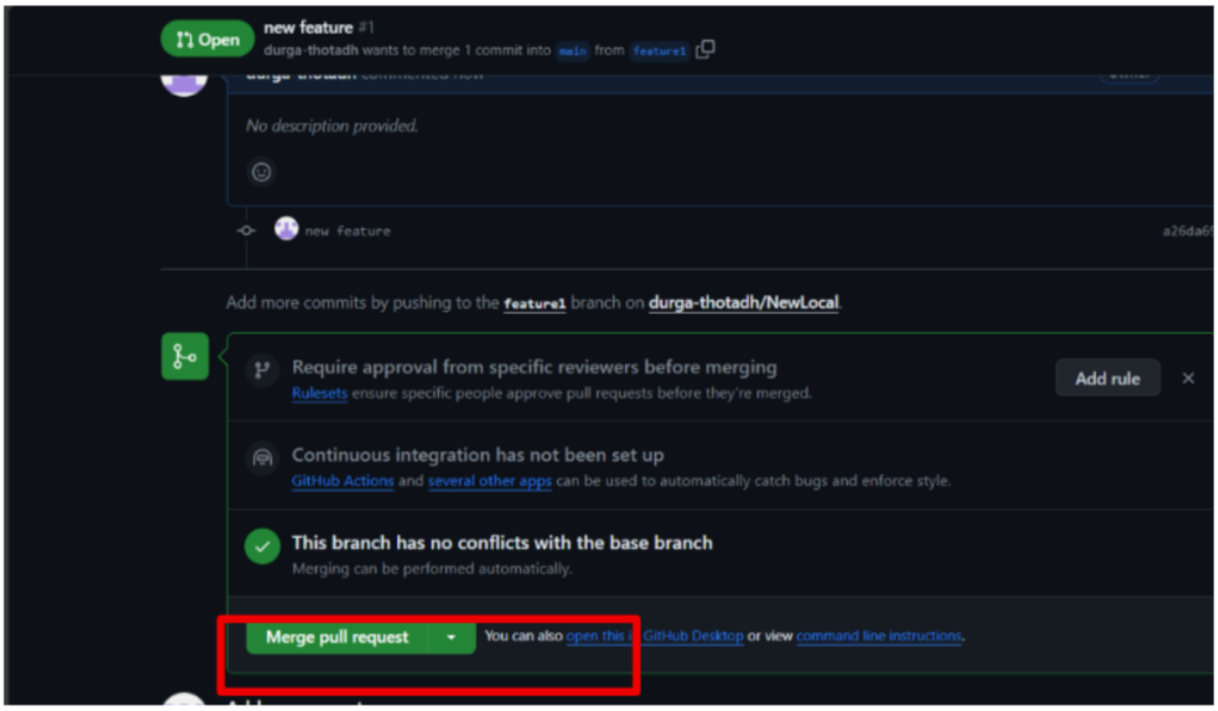
3. Above the list of files, in the yellow banner, click **Compare & pull request** to create a pull request for the associated branch.
4. Type a **title and description** for your pull request.
5. To create a pull request that is ready for review, click **Create Pull**

Request.

3.5 Merging a Pull request

1. Under your repository name, click **Pull requests**.
2. In the "Pull Requests" list, click the pull request you'd like to merge.
3. Scroll down to the bottom of the pull request.

4. Depending on the merge options enabled for your repository: Merge all of the commits into the base branch by clicking **Merge pull request**.
5. If the **Merge pull request** option is not shown, click the merge dropdown menu and select **Create a merge commit**.



new feature #1
durga-thotadh wants to merge 1 commit into main from feature1

No description provided.

new feature

Add more commits by pushing to the feature1 branch on durga-thotadh/NewLocal.

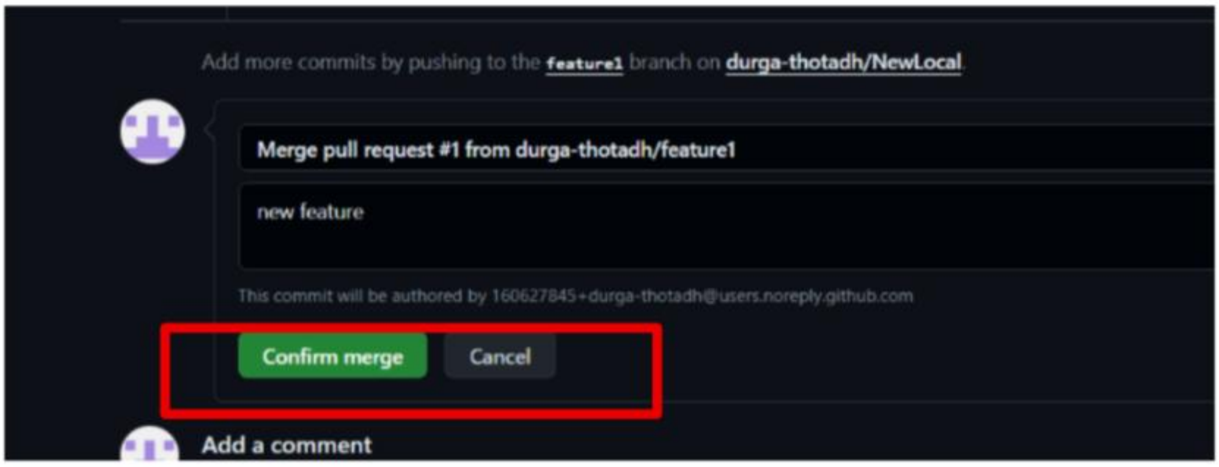
Require approval from specific reviewers before merging
Rulesets ensure specific people approve pull requests before they're merged.

Continuous integration has not been set up
GitHub Actions and several other apps can be used to automatically catch bugs and enforce style.

This branch has no conflicts with the base branch
Merging can be performed automatically.

You can also open this i GitHub Desktop or view command line instructions.

Figure: selecting a pull request to merge



Add more commits by pushing to the feature1 branch on durga-thotadh/NewLocal

Merge pull request #1 from durga-thotadh/feature1

new feature

This commit will be authored by 160627845+durga-thotadh@users.noreply.github.com

Figure: confirming merge pull request

6. Click **Confirm merge**,

5.GIT COMMANDS:

1: “git status”

This simple command provides a number of details about the current branch. These details include the update status of the branch and whether or not there’s anything to commit, push, or pull. This also shows any other changes that have been staged, such as files staged or files created/modified/deleted.

2: “git show”

Usage: git show [commit]

This command is used to display metadata for a specified commit.

3: “git init”

Usage: git init [repository name]

This is the command for starting a new repository.

4: “git commit”

Usage: git commit -m “[Type in the commit message]”

Use the “commit” command to save a copy of the file in the version history of the project.

5: “git branch”

Usage: git branch

When your command line is located in a particular repository, you can use “git branch” to see a list of all the local branches in that current repository.

6: “git add”

Usage: git add [file]

A command for adding a file to the staging area.

7: “git pull”

Usage: git pull [repository link]

This command is a vital tool for remote collaboration. You can use “git pull” to bring changes to the file on the remote server into your current working directory and merge them with your local project.

8: “git push”

Usage: git push [variable name] master / git push [variable name] branch / git push -all [variable name] / git push [variable name] :[branch name]

Git push is the command developers use to send their local changes to the remote repository. Depending on which version of the usage you utilize, you can commit just the current branch, or the entire master branch to your remote repository. You can also push all branches, or even delete a branch using this command.

9: “git revert”

Usage: git revert [hash code of the specific commit to be undone]

A command that can be used in conjunction with “git log” to find a specific change (commit) and remove it from the repository. A method of undoing a mistake.

10: “git log”

Usage: git log

A straightforward method to display the version history for the current

branch.

5.REFERENCES:

1 . https://www.simplilearn.com/tutorials/git-tutorial/what-is-git What is Git: Features, Commands, Branch and Workflow in Git (simplilearn.com)

2. https://www.freecodecamp.org/news/what-is-git-and-how-to-use-it-c341b049ae61/ An introduction to Git: what it is, and how to use it (freecodecamp.org)

3. https://opensource.com/article/18/1/step-step-guide-git A step-by-step guide to Git | Opensource.com

4. https://www.linkedin.com/pulse/git-blog-2-features-demonstration-james-forrester (9) Git Blog 2: Features and Demonstration | LinkedIn

5. https://devmountain.com/blog/top-git-commands-how-to-use/ Top Git Commands | How to Use Git Commands | Devmountain