



Experiment -1.2

Creating branches with GitHub

Student Name: SAKSHI

Branch: CSE(DevOps)

Semester: 4th

Subject Name: Git and GitHub

UID: 22BDO10064

Section/Group: 22BCD-1/B

Date of Performance: 19/01/24

Subject Code: 22CSH-293

1. Aim/Overview of the practical: Creating branches with GitHub.

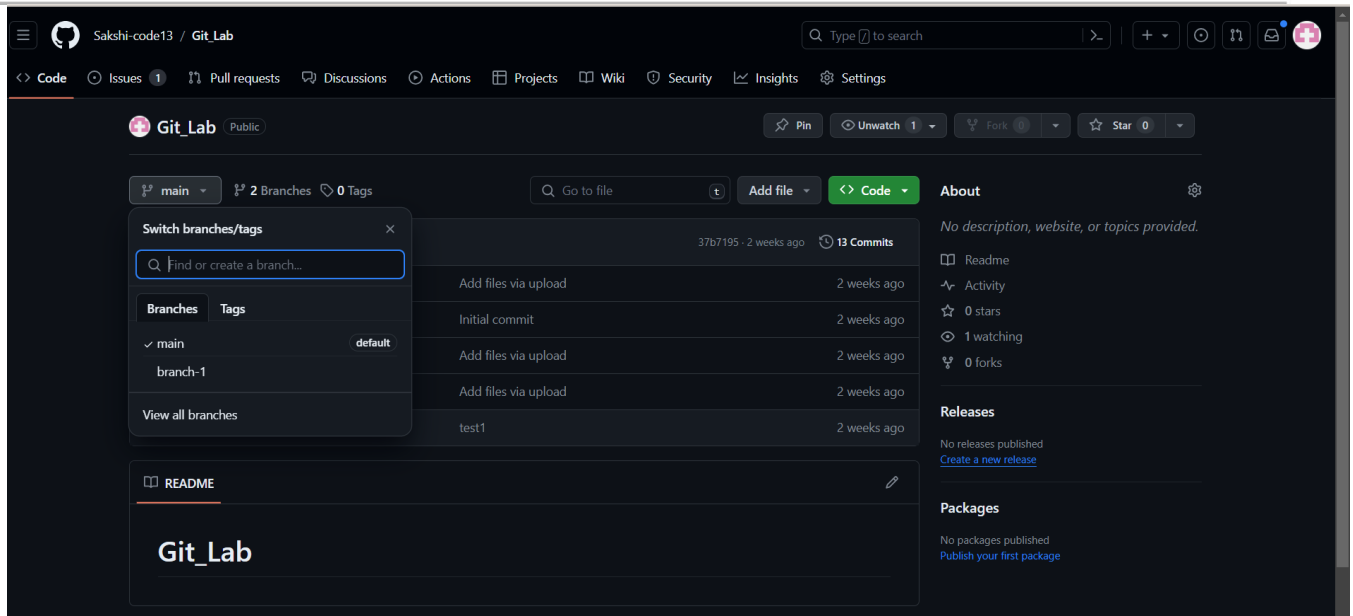
2. Task to be done: In this experiment, we have to create a branch, merging a branch and deleting a branch.

3. Steps for experiment/practical:

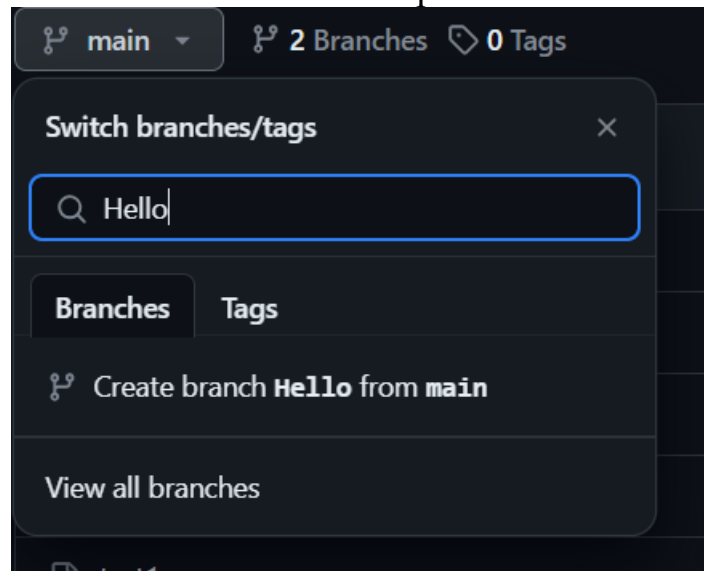
a) By using GitHub / GUI :-

1. Create a New Branch:

- First of all, login to your GitHub account.
- Navigate to your repository.



- Click on to the “Main:branch” button which is present at the left of the page.



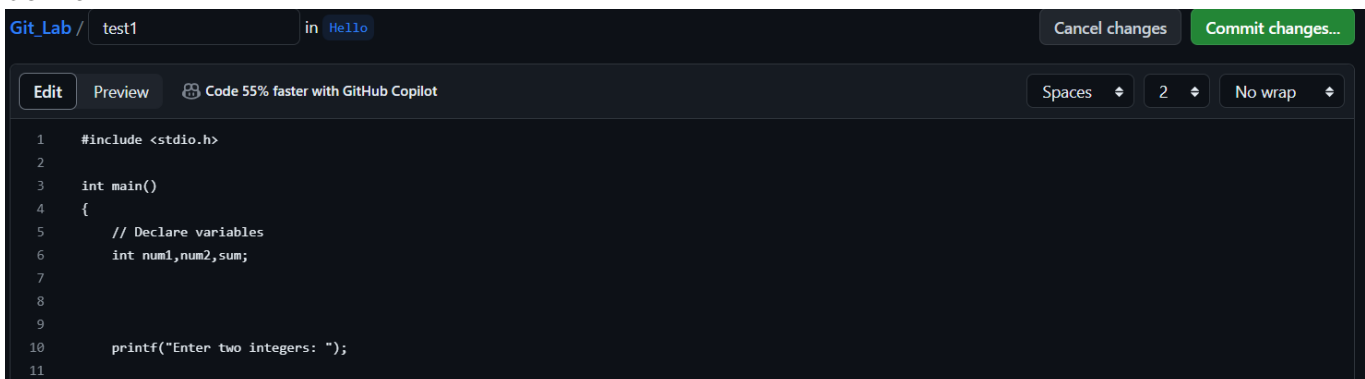
- For creating a new branch type a name for your branch and press enter.
- You can select the branch from clicking the dropdown menu.
- GitHub Account → “Branch:main” button → Type a new name for your new branch → press Enter.

2. Make Changes:

- There are two options present for making changes: - 1. Make changes to your code 2. Add new files
- Go to the dropdown menu of the branch → Select the branch you want to make the changes.

3. Edit and Commit Changes:

- For committing the changes, press the commit button which is present at the upper right corner

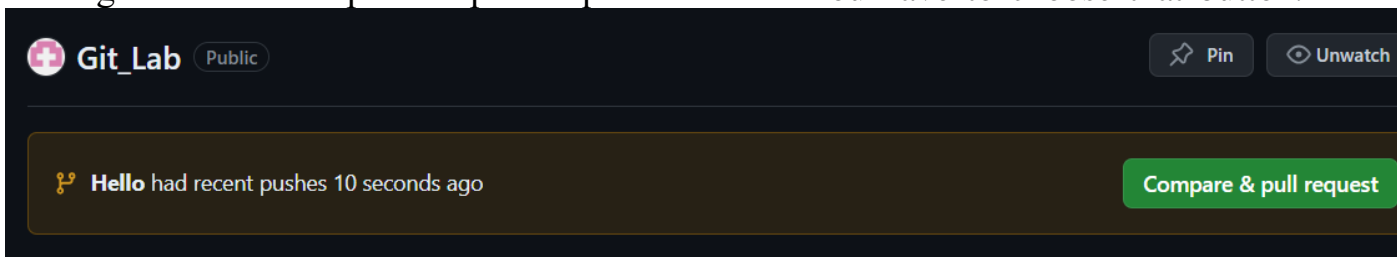


```
1  #include <stdio.h>
2
3  int main()
4  {
5      // Declare variables
6      int num1, num2, sum;
7
8
9
10     printf("Enter two integers: ");
11
```

- Make changes to your code → Scroll down to the changes section → Click on the “Commit changes” button.

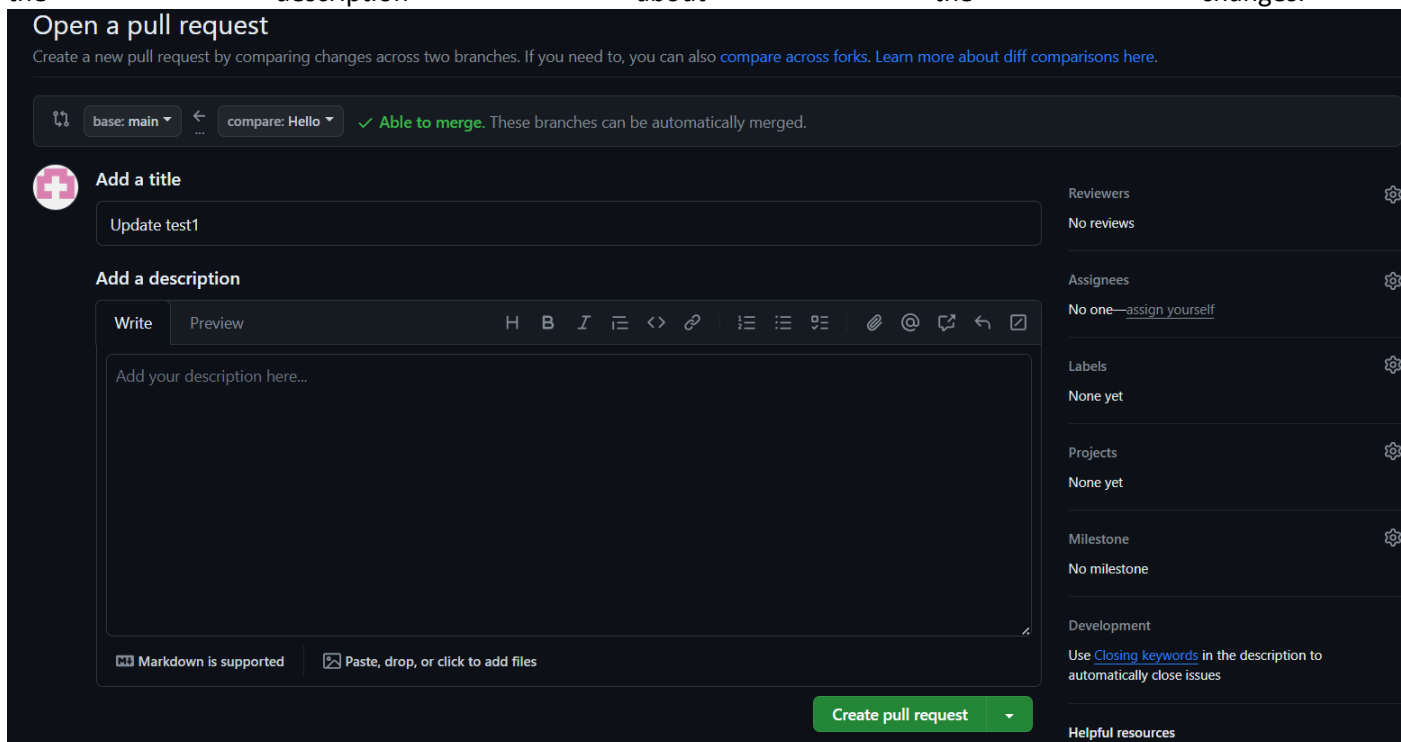
4. Create a Pull Request:

- Now, once you’ve committed the changes to the new branch, GitHub will display a message with a “Compare & pull request” button. You have to choose that button.



5. Open a Pull Request:

- For opening a pull request, you'll be taken to a page where you can review your changes. Also you can add the description about the changes.



The screenshot shows the GitHub 'Open a pull request' page. At the top, it says 'Open a pull request' and 'Create a new pull request by comparing changes across two branches. If you need to, you can also [compare across forks](#). [Learn more about diff comparisons here](#).' Below this, there are two dropdown menus: 'base: main' and 'compare: Hello'. A green checkmark indicates 'Able to merge. These branches can be automatically merged.' The main section is titled 'Add a title' and contains a text input field with 'Update test1'. Below that is 'Add a description' with a 'Write' tab selected and a rich text editor. The editor has a toolbar with various formatting options and a large text area with the placeholder 'Add your description here...'. At the bottom of the editor, it says 'Markdown is supported' and 'Paste, drop, or click to add files'. On the right side, there are several sections: 'Reviewers' (No reviews), 'Assignees' (No one—[assign yourself](#)), 'Labels' (None yet), 'Projects' (None yet), 'Milestone' (No milestone), and 'Development' (Use [Closing keywords](#) in the description to automatically close issues). At the bottom right, there is a green 'Create pull request' button and a 'Helpful resources' link.

6. Merge the Pull Request:

- When your pull request is reviewed and approved, you can merge it.
- Click on the “Merge pull request” button.

Update test1 #10

Open Sakshi-code13 wants to merge 1 commit into [main](#) from [Hello](#)

Conversation 0 Commits 1 Checks 0 Files changed 1

Sakshi-code13 commented 9 minutes ago Owner

No description provided.

Update test1 Verified 97dd6a2

Add more commits by pushing to the [Hello](#) branch on [Sakshi-code13/Git_Lab](#).

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

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.

Merge pull request You can also [open this in GitHub Desktop](#) or view [command line instructions](#).

Reviewers
No reviews
Still in progress? [Convert to draft](#)

Assignees
No one—[assign yourself](#)

Labels
None yet

Projects
None yet

Milestone
No milestone

Development
Successfully merging this pull request may close these issues.
None yet

7. Confirm the Merge:

- When you merge the pull requests there'll come a message for confirming merge.

Add more commits by pushing to the [Hello](#) branch on [Sakshi-code13/Git_Lab](#).

Merge pull request #10 from Sakshi-code13/Hello

Update test1

This commit will be authored by 119587392+Sakshi-code13@users.noreply.github.com

Confirm merge **Cancel**

8. Delete the Branch:

- The option of Deleting the branch is also available on GitHub after merging it. So that there exists no confusion regarding the older branch. So, you can choose the branch if you want to.

Update test1 #10

Merged Sakshi-code13 merged 1 commit into `main` from `Hello` now

Conversation 0 Commits 1 Checks 0 Files changed 1

Sakshi-code13 commented 14 minutes ago Owner ...

No description provided.

Update test1 Verified 97dd6a2

Sakshi-code13 merged commit 350f079 into `main` now Revert

Pull request successfully merged and closed Delete branch

You're all set—the `Hello` branch can be safely deleted.

9. Restore the branch:

- After deleting the branch, you can also restore it. As there is one option present regarding the restore of the branch. So, if you want you can restore it.

Sakshi-code13 deleted the `Hello` branch now Restore branch

b) By Using Git Bash:-

1. Create a new file:

```
MINGW64:/c/Users/ADMIN  
  
ADMIN@LAPTOP-RFULERP MINGW64 ~  
$ touch Sakshi.txt
```

- For creating a file the command is used **\$ touch file_name.txt**

2. Initialize a Git repository:

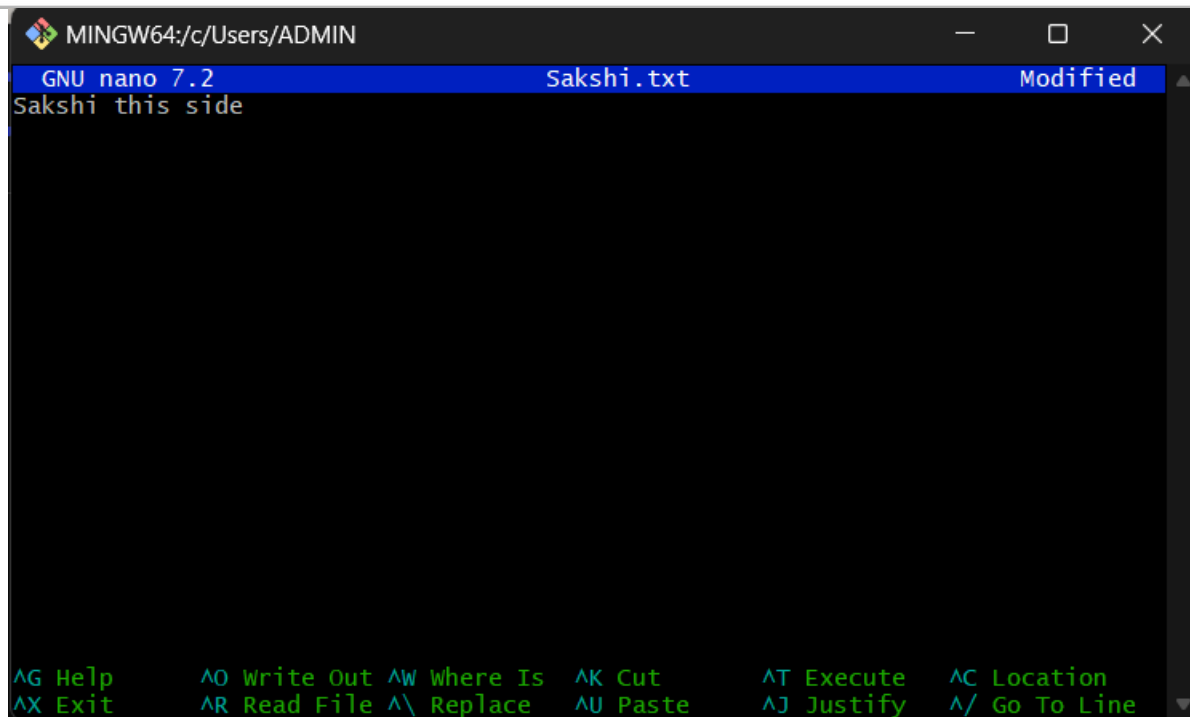
```
ADMIN@LAPTOP-RFULERP MINGW64 ~  
$ git init  
Initialized empty Git repository in C:/Users/ADMIN/.git/
```

- For initializing an empty git repository in the current working directory use the command **\$ git init**

3. Open and edit a file:

```
MINGW64:/c/Users/ADMIN  
  
ADMIN@LAPTOP-RFULERP MINGW64 ~ (master)  
$ nano Sakshi.txt
```

- For editing a file you can use the command **\$ nano file_name.txt**
- Now a textbox will prompt up where you can make the changes, and to exit from there just press **Cntrl + X**.



```
MINGW64:/c/Users/ADMIN
GNU nano 7.2 Sakshi.txt Modified
Sakshi this side

^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute   ^C Location
^X Exit      ^R Read File ^\ Replace   ^U Paste     ^J Justify   ^_ Go To Line
```

4. Add the changes to the staging area:

- To see the changes that are made into the file or to add the changes to the staging area, you can use the command **\$ cat file_name.txt**

```
ADMIN@LAPTOP-RFULERMP MINGW64 ~ (master)
$ cat Sakshi.txt
Sakshi this side
```

5. Commit changes to the main branch:

- To commit the change in the staging area with an appropriate commit message to the main branch you can use the following commands:-
- \$ git add file_name.txt**
- By using this command a warning will be printed on the screen.
- Now, enter the next command, **\$ git commit -m "Message you want to print at successful commit"**.


```
ADMIN@LAPTOP-RFULERP MINGW64 ~ (master)
$ git add Sakshi.txt
warning: in the working copy of 'Sakshi.txt', LF will be replaced by CRLF the ne
xt time Git touches it

ADMIN@LAPTOP-RFULERP MINGW64 ~ (master)
$ git commit -m "Committed Successfully"
[master (root-commit) c35cf33] Committed Successfully
1 file changed, 1 insertion(+)
create mode 100644 Sakshi.txt
```

6. Create a new branch:

- For switching to a new branch, you can use the command: `$ git checkout -b newBranch` name

```
ADMIN@LAPTOP-RFULERP MINGW64 ~ (master)
$ git checkout -b testBranch
Switched to a new branch 'testBranch'
```

7. Open and edit files in the new branch:

- Now, you can edit the file, which is now shifted on the newBranch.
- By using the same command to normally edit into a file `$ nano Sakshi.txt`.
- A text box will prompt where you can make the desired changes.
- And, then to see the changes you can use the command `$ cat Sakshi.txt`.

```
ADMIN@LAPTOP-RFULERP MINGW64 ~ (testBranch)
$ nano Sakshi.txt

ADMIN@LAPTOP-RFULERP MINGW64 ~ (testBranch)
$ cat Sakshi.txt
Sakshi this side
How are you doing ?
```

8. Add changes to the staging area in the new branch:

- Add all the changes made into the working directory to the staging area.

9. Commit changes to the new branch:

- Now commit all the changes to the new branch.
- By using the command `$ vi file_name.txt`

- \$ git add
- \$ git commit -m "Made the changes successfully!!!"

```
ADMIN@LAPTOP-RFULERP MINGW64 ~ (master)
$ git add Sakshi.txt
warning: in the working copy of 'Sakshi.txt', LF will be replaced by CRLF the next time Git touches it

ADMIN@LAPTOP-RFULERP MINGW64 ~ (master)
$ git commit -m "Committed Successfully"
[master (root-commit) c35cf33] Committed Successfully
1 file changed, 1 insertion(+)
create mode 100644 Sakshi.txt
```

10. Switch back to the main branch:

- Switch to the main branch use the command \$ **git checkout branch_name**
- Also, you can switch the branches by using the command \$ **git switch branch_name**

```
ADMIN@LAPTOP-RFULERP MINGW64 ~ (testBranch)
$ git checkout master
Switched to branch 'master'
```

```
ADMIN@LAPTOP-RFULERP MINGW64 ~ (testBranch)
$ git switch master
Switched to branch 'master'
```

11. Merge changes from the new branch to main:

- We can merge the branches by using the command \$ **git merge branch_name**

```
ADMIN@LAPTOP-RFULERP MINGW64 ~ (master)
$ git merge testBranch
Updating c35cf33..e71634a
Fast-forward
 Sakshi.txt | 1 +
1 file changed, 1 insertion(+)
```

12. To list all the branches:

- To list all the local branches, run the following command : **\$ git branch**

```
MINGW64:/c/Users/ADMIN
ADMIN@LAPTOP-RFULERP MINGW64 ~ (testBranch)
$ git branch
master
* testBranch
testBranch1
```

- To list all the remote branches, run the following command: **\$ git branch -r**
- To list all the remote as well as local branches, run the following command: **\$ git branch -a**

```
MINGW64:/c/Users/ADMIN
ADMIN@LAPTOP-RFULERP MINGW64 ~ (master)
$ git branch -r
ADMIN@LAPTOP-RFULERP MINGW64 ~ (master)
$ git branch -a
* master
testBranch
testBranch1
```

13. To delete a branch, \$ git branch -d branch_name is used.

```
ADMIN@LAPTOP-RFULERP MINGW64 ~ (master)
$ git branch -d testBranch1
Deleted branch testBranch1 (was c35cf33).
```

```
ADMIN@LAPTOP-RFULERP MINGW64 ~ (master)
$ git branch -d testBranch
Deleted branch testBranch (was c35cf33).
```

```
ADMIN@LAPTOP-RFULERP MINGW64 ~ (master)
$ git branch
* master
```

14. Check the status of the repository: For verifying that there's nothing left behind to clean and commit in the main branch. Use the command **\$ git status**

4. Result:

In this experiment, we have explored two distinct methodologies for creating branches in the repositories. The first approach entails utilizing a graphical user interface (GUI) by directly working on GitHub. We initiated the creation of a branch, added specific files to it, performed a meticulous comparison, and subsequently merged it with the main branch. Additionally, we adeptly managed the deletion of said branch. Contrastingly, the second approach necessitated the utilization of Git Bash, where we proficiently employed a multitude of commands to accomplish the desired outcome.

Learning outcomes (What I have learnt):

1. Understanding Git Workflow
2. Different approaches i.e., Git Bash, GUI
3. Branches and its functionalities.
4. Various commands to handle different function related to the branches.
5. Working with and on the repositories.

Evaluation Grid (To be created as per the SOP and Assessment guidelines by the faculty):

Sr. No.	Parameters	Marks Obtained	Maximum Marks
1.			
2.			
3.			