



**AMITY INSTITUTE OF
INFORMATION AND TECHNOLOGY**

Course Code: CSE2015

“SOURCE CODE MANAGEMENT”

LABORATORY RECORD

Winter Semester

2024 - 25

SUBMITTED TO:

**Dr. Monit Kapoor
Pro Vice Chancellor
Amity University Bengaluru**

SUBMITTED BY:

**Pavan Kumar B
A866185824005
Amity University Bengaluru**

INDEX

| SL.NO | DATE | EXERCISE |
|-------|------------|---|
| 01 | 03/02/2025 | Setting up Git Environment and Basic Configuration |
| 02 | 10/02/2025 | Creating and Managing Local Repositories |
| 03 | 17/02/2025 | Working with Basic Git Commands (add, commit, status) |
| 04 | 24/02/2025 | Managing Files and Directories in Git |
| 05 | 10/03/2025 | Understanding Git Workflow and Staging Area |
| 06 | 24/03/2025 | Creating and Switching Branches |
| 07 | 07/04/2025 | Merging branches and resolving conflicts |
| 08 | 21/04/2025 | Working with Remote Repositories |
| 09 | 05/05/2025 | Collaborating using Pull Requests |

Lab Exercise 1: Setting up Git Environment and Basic Configuration

Objective

To install Git version control system and configure basic user settings for version control operations.

Prerequisites

1. Operating System (Windows/Linux/macOS)
2. Administrator/sudo privileges
3. Internet connection for installation

Environment Details

1. Operating System: windows 11
2. Git Version: [Version Number from git --version]

Implementation Steps

1. Git Installation

1. open your browser
2. search for official website for downloading git (git bash, git gui)
3. click on windows link to download
4. once download is complete run the installer.
5. choose the git components you want to install.
6. click next to complete the installation.

2. Installation Verification:

- Command:

```
git --version
```

- Output:

```
HP eh 1047au@TARUN MINGW64 ~  
$ git --version  
git version 2.49.0.windows.1  
  
HP eh 1047au@TARUN MINGW64 ~  
$ |
```

3. Project Setup: Command `mkdir scm_lab_01` `cd`

`scm_lab_01`

`git init`

Output:

```
HP eh 1047au@TARUN MINGW64 ~  
$ mkdir scm_lab_01  
  
HP eh 1047au@TARUN MINGW64 ~  
$ pwd  
/c/Users/HP eh 1047au  
  
HP eh 1047au@TARUN MINGW64 ~  
$ cd scm_lab_01  
  
HP eh 1047au@TARUN MINGW64 ~/scm_lab_01  
$ git init  
Initialized empty Git repository in C:/Users/HP eh 1047au/scm_lab_01/.git/  
  
HP eh 1047au@TARUN MINGW64 ~/scm_lab_01 (master)  
$ |
```

4. Git Configuration

Commands

Git config user.name "Akash"

Git config "user.email "

- Verification commands and expected output :

Git config user.name

Git config user.email

```
HP eh 1047au@TARUN MINGW64 ~/scm_lab_01
$ git init
Initialized empty Git repository in C:/Users/HP eh 1047au/scm_lab_01/.git/

HP eh 1047au@TARUN MINGW64 ~/scm_lab_01 (master)
$ git config --global user.name "Prathish0169"

HP eh 1047au@TARUN MINGW64 ~/scm_lab_01 (master)
$ git config --global user.email "prathish.achar@s.amity.edu"

HP eh 1047au@TARUN MINGW64 ~/scm_lab_01 (master)
$ |
```

Challenges Faced

- [Document any issues encountered during the setup and how they were resolved]

Learning Outcomes

1. Understanding of Git installation process
2. Knowledge of basic Git configuration
3. Ability to verify Git setup and configuration
4. Familiarity with Git command-line interface

Conclusion

Configured Git for streamlined source code management, ensuring efficient tracking, collaboration, and version control in development projects.

Lab exercise 2: Creating and managing local repositories

Objective:

To create local repository and understand its basic structure and initialization process.

Perequisites:

- 1.git
- 2.basic command line familiarity
- 3.project directory for familiarity

Environment details:

- 1.git version: 2.47.1windows.2

2.working directory:/c/users/maha/downloads/scm_lab2

Implementation steps:

1.Respository creation

#commands

Git init

Output

```
MINGW64:/c/Users/HP eh 1047au/scm_lab_01
HP eh 1047au@TARUN MINGW64 ~/scm_lab_01 (master)
$ git init
Reinitialized existing Git repository in C:/Users/HP eh 1047au/scm_lab_01/.git/
HP eh 1047au@TARUN MINGW64 ~/scm_lab_01 (master)
$
```

2.respository structure analysis: #commands

cd.git ls-al

output

```
HP eh 1047au@TARUN MINGW64 ~/scm_lab_01 (master)
$ git init
Reinitialized existing Git repository in C:/Users/HP eh 1047au/scm_lab_01/.git/
HP eh 1047au@TARUN MINGW64 ~/scm_lab_01 (master)
$ cd .git
HP eh 1047au@TARUN MINGW64 ~/scm_lab_01/.git (GIT_DIR!)
$ ls -al
total 11
drwxr-xr-x 1 HP eh 1047au 197121  0 May 31 22:26 ./
drwxr-xr-x 1 HP eh 1047au 197121  0 May 31 22:21 ../
-rw-r--r-- 1 HP eh 1047au 197121  23 May 31 22:21 HEAD
-rw-r--r-- 1 HP eh 1047au 197121 130 May 31 22:26 config
-rw-r--r-- 1 HP eh 1047au 197121  73 May 31 22:21 description
drwxr-xr-x 1 HP eh 1047au 197121  0 May 31 22:21 hooks/
drwxr-xr-x 1 HP eh 1047au 197121  0 May 31 22:21 info/
drwxr-xr-x 1 HP eh 1047au 197121  0 May 31 22:21 objects/
drwxr-xr-x 1 HP eh 1047au 197121  0 May 31 22:21 refs/
HP eh 1047au@TARUN MINGW64 ~/scm_lab_01/.git (GIT_DIR!)
$
```

4.respository status check

#commands

Git status

#expected output

Git status

#expected output

```
HP eh 1047au@TARUN MINGW64 ~/scm_lab_01/.git (GIT_DIR!)
$ git status
fatal: this operation must be run in a work tree

HP eh 1047au@TARUN MINGW64 ~/scm_lab_01/.git (GIT_DIR!)
$ |
```

Challenges faced :

No challenges faced

Learning outcomes

- 1.understanding of git repository intialization
- 2.Knowlodge of git directory structure
- 3.ability to verify repository status
- 4.Recognition of clean/empty repository status

Conclusion:

1.successfully created and examined the structure of the local git repository establishing the foundation for version control operations

Lab exercise 3: Working with basic git commands (add,commit,status)

Objective:

To understand and practice the basic git commands for staging and committing changes while monitoring repository status

Prerequisites:

- 1.Git repository initialization
- 2.Basic understanding of git directory structure

Environment details:

- 1.Git version : 2.47.1.windows.2
- 2.Working directory: /c/Users/Maha/downloads/scm_lab3

Implementation steps :

1. Creating a sample file

#commands

1. echo "This is my first file." > 1.txt

2. ls -al

#output

```
HP eh 1047au@TARUN MINGW64 ~/scm_lab_01/.git (GIT_DIR!)
$ echo "this is my first file.">1.txt

HP eh 1047au@TARUN MINGW64 ~/scm_lab_01/.git (GIT_DIR!)
$ ls -al
total 12
drwxr-xr-x 1 HP eh 1047au 197121  0 May 31 22:29 ./
drwxr-xr-x 1 HP eh 1047au 197121  0 May 31 22:21 ../
-rw-r--r-- 1 HP eh 1047au 197121 23 May 31 22:29 1.txt
-rw-r--r-- 1 HP eh 1047au 197121 23 May 31 22:21 HEAD
-rw-r--r-- 1 HP eh 1047au 197121 130 May 31 22:26 config
-rw-r--r-- 1 HP eh 1047au 197121  73 May 31 22:21 description
drwxr-xr-x 1 HP eh 1047au 197121  0 May 31 22:21 hooks/
drwxr-xr-x 1 HP eh 1047au 197121  0 May 31 22:21 info/
drwxr-xr-x 1 HP eh 1047au 197121  0 May 31 22:21 objects/
drwxr-xr-x 1 HP eh 1047au 197121  0 May 31 22:21 refs/

HP eh 1047au@TARUN MINGW64 ~/scm_lab_01/.git (GIT_DIR!)
$
```

2. Checking repository status:

#commands

1. git status

#output

3. Adding file to staging area:

```
HP eh 1047au@TARUN MINGW64 ~/scm_lab_01/.git (GIT_DIR!)
$ git status
fatal: this operation must be run in a work tree

HP eh 1047au@TARUN MINGW64 ~/scm_lab_01/.git (GIT_DIR!)
$
```

#command

1.git add test.txt

2.git status

#output

4.Adding multiple files:

#commands

1.echo "This is my 2nd file" >2.txt 2.echo "This is my 3rd file " >3.txt

3.git add .

4.git status

#output

```
P eh 1047au@TARUN MINGW64 ~/scm_lab_01/.git (GIT_DIR!)
vi hello.cpp

P eh 1047au@TARUN MINGW64 ~/scm_lab_01/.git (GIT_DIR!)
git add .
fatal: this operation must be run in a work tree

P eh 1047au@TARUN MINGW64 ~/scm_lab_01/.git (GIT_DIR!)
git status
fatal: this operation must be run in a work tree

P eh 1047au@TARUN MINGW64 ~/scm_lab_01/.git (GIT_DIR!)

```

5.Creating initial commit: #commands git commit -m "initial commit"

#output

```
commit

HP eh 1047au@TARUN MINGW64 /c/aboutme (master)
$ git commit -m "file is created"
[master (root-commit) 87773a9] file is created
1 file changed, 6 insertions(+)
create mode 100644 hello.cpp

HP eh 1047au@TARUN MINGW64 /c/aboutme (master)
$

```

6.Modifying files:

#commands

1.echo "I AM MODIFYING THIS FILE. >> test.txt

2.echo "I AM sukruth" >> test1.txt

```
HP eh 1047au@TARUN MINGW64 /c/aboutme (master)
$ echo "i am modifiedthis file".>>test.txt

HP eh 1047au@TARUN MINGW64 /c/aboutme (master)
$ echo "i am prathish".>> test1.txt

HP eh 1047au@TARUN MINGW64 /c/aboutme (master)
$ |
```

4.git status

#output

```
HP eh 1047au@TARUN MINGW64 /c/aboutme (master)
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    test.txt
    test1.txt

nothing added to commit but untracked files present (use "git add" to track)
HP eh 1047au@TARUN MINGW64 /c/aboutme (master)
$ |
```

7.Viewing changes:

#commands

1.git diff

#output

```
diff --git a/test.txt b/test.txt
index 61aae3e..0c3479c 100644
--- a/test.txt
+++ b/test.txt
@@ -1,2 @@
- this is my file
+ i am modifying this file.
diff --git a/test1.txt b/test1.txt
: ...skipping...
warning: in the working copy of 'test.txt', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'test1.txt', LF will be replaced by CRLF the next time Git touches it
diff --git a/test.txt b/test.txt
index 61aae3e..0c3479c 100644
--- a/test.txt
+++ b/test.txt
```

8.Adding and committing modified files:

#commands

1.git add .

2.git commit -m "updated files"

#output

```
$ git commit -m "updated files "  
[master f53ac60] updated files  
3 files changed, 3 insertions(+)
```

Challenges faced: No challenges faced **Learning**

outcomes:

- 1.Understanding of staging area concept
- 2.Mastery of git commands (add,commit,status)
- 3.Ability to track file changes
- 4.Knowledge of commit message best practices
- 5.Understanding difference between tracked and untracked file

Conclusion:

Successfully learned and implemented basic git commands for staging and committing changes establishing fundamental version control workflow

Lab exercise 4: Managing files and directories in git

Objective:

To understand and practice git commands for managing files and directories ,including adding, tracking modifications, removing and renaming files

Prerequisites:

- 1.Git repository initialization
- 2.Basic understanding of git commands(add,commit,status)
- 3.Existing repository with previous commits

Environment details:

1. Git version: 2.47.1.windows.2
- 2.Working directory: /c/Users/Maha/downloads/scm_lab4

Implementation steps:

1.Adding new files to repository:

#Commands

- 1.cd downloads
- 2.mkdir scm_lab4
- 3.cd scm_lab4
- 4.git init
- 5.echo "This is my 1 st file " > 1.txt
6. echo "This is my 2nd file " > 2.txt
7. echo "This is my 3 rd file " > 3.txt
- 8.git status
- 9.git add .
- 10.git status
- 11.git commit -m "add 1.txt,2.txt,3.txt"

#Output

```
HP eh 1047au@TARUN MINGW64 /c/scm_lab_01
$ cd /c

HP eh 1047au@TARUN MINGW64 /c
$ mkdir aboutme

HP eh 1047au@TARUN MINGW64 /c
$ cd aboutme

HP eh 1047au@TARUN MINGW64 /c/aboutme
$ vi hello.cpp

HP eh 1047au@TARUN MINGW64 /c/aboutme
$ git init
Initialized empty Git repository in C:/aboutme/.git/

HP eh 1047au@TARUN MINGW64 /c/aboutme (master)
$ git add .
warning: in the working copy of 'hello.cpp', LF will be replaced by CRLF the next time Git touches it
```

```
HP eh 1047au@TARUN MINGW64 /c/aboutme (master)
$ git commit -m "commit first"
[master 418d3b5] commit first
3 files changed, 3 insertions(+)
create mode 100644 test.txt
create mode 100644 test1.txt

HP eh 1047au@TARUN MINGW64 /c/aboutme (master)
$ git status
On branch master
nothing to commit, working tree clean

HP eh 1047au@TARUN MINGW64 /c/aboutme (master)
```

```
HP eh 1047au@TARUN MINGW64 /c/aboutme (master)
$ git commit -m "commit first"
[master 418d3b5] commit first
3 files changed, 3 insertions(+)
create mode 100644 test.txt
create mode 100644 test1.txt
```

2.Tracking modifications:

#Commands

- 1.echo "I am good" >> 1.txt
- 2.echo " I am handsome " >> 2.txt
3. echo " I am a good boy " >> 3.txt
- 4.git status
- 5.git diff
- 6.git add .
- 7.git status
- 8.git commit – m " add modified files 1.txt,2.txt,3.txt

#Output

```

HP eh 1047au@TARUN MINGW64 /c/aboutme (master)
$ echo " I am good ">> test.txt

HP eh 1047au@TARUN MINGW64 /c/aboutme (master)
$ echo " I am bad">> test1.txt

HP eh 1047au@TARUN MINGW64 /c/aboutme (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   test1.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        test.txt

no changes added to commit (use "git add" and/or "git commit -a")

HP eh 1047au@TARUN MINGW64 /c/aboutme (master)
$

```

```

HP eh 1047au@TARUN MINGW64 /c/aboutme (master)
$ git diff
warning: in the working copy of 'test1.txt', LF will be replaced by CRLF the next time Git touches it
diff --git a/test1.txt b/test1.txt
index cc4a943..b11c73d 100644
--- a/test1.txt
+++ b/test1.txt
@@ -1,2 @@
 i am prathish.
+ I am bad

HP eh 1047au@TARUN MINGW64 /c/aboutme (master)
$

```

3.Adding all changes at once:

#Commands

#Output

4.Removing files from repository:

#Commands

1.rm 1.txt 2.txt

2.git status

3.git rm 1.txt 2.txt

4.git commit -m "remove files 1.txt ,2.txt"

#Output

```
HP eh 1047au@TARUN MINGW64 ~/pratathish2 (new-branch-name)
$ rm test.txt test1.txt
rm: cannot remove 'test.txt': No such file or directory
rm: cannot remove 'test1.txt': No such file or directory

HP eh 1047au@TARUN MINGW64 ~/pratathish2 (new-branch-name)
$ ls
README.md  'README.md'$'\n'

HP eh 1047au@TARUN MINGW64 ~/pratathish2 (new-branch-name)
$ git status
On branch new-branch-name
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    README.md\357\200\212"

nothing added to commit but untracked files present (use "git add" to track)

HP eh 1047au@TARUN MINGW64 ~/pratathish2 (new-branch-name)
$ |
```

5.Removing files from staging area only:

#Commands

- 1.touch 4.txt
- 2.touch 5.txt
- 3.touch 6.txt
- 4.git add .
- 5.git status
- 6.git restore – staged 4.txt 5.txt
- 7.git status

Lab Exercise 5: Understanding Git Workflow and Staging Area Objective:

To understand and practice Git's three-stage workflow, staging area functionality, and tracking file states throughout the development process.

Prerequisites:

- Git repository initialization
- Basic understanding of Git commands (init, add, commit)
- Command line interface proficiency

Environment Details:

- Git Version: git version 2.39.5 (Apple Git-154)
- Working Directory: /Users/Prathish/Desktop/git-demo/.git/

Implementation Steps:

1. Git Data Model Exploration:

#Commands

1.cd desktop

2.mkdir git demo

3.cd git-demo

4.pwd

5.ls -al

6.git init

7.nano hello.txt

8.cat hello.txt

9.ls -al

#Output

```
Reinitialized existing Git repository in C:/Users/HP eh 1047au/destop/.git/
HP eh 1047au@TARUN MINGW64 ~/destop (master)
$ mkdir git-demo
mkdir: cannot create directory 'git-demo': File exists

HP eh 1047au@TARUN MINGW64 ~/destop (master)
$ mkdir git-demo1

HP eh 1047au@TARUN MINGW64 ~/destop (master)
$ pwd
/c/Users/HP eh 1047au/destop

HP eh 1047au@TARUN MINGW64 ~/destop (master)
$ ls -al
total 32
drwxr-xr-x 1 HP eh 1047au 197121 0 Jun  2 18:02 ./
drwxr-xr-x 1 HP eh 1047au 197121 0 Jun  2 18:02 ../
drwxr-xr-x 1 HP eh 1047au 197121 0 Jun  2 18:02 .git/
drwxr-xr-x 1 HP eh 1047au 197121 0 Jun  2 16:32 git-demo/
drwxr-xr-x 1 HP eh 1047au 197121 0 Jun  2 18:02 git-demo1/

HP eh 1047au@TARUN MINGW64 ~/destop (master)
$ git init
Reinitialized existing Git repository in C:/Users/HP eh 1047au/destop/.git/

HP eh 1047au@TARUN MINGW64 ~/destop (master)
$ cd git-demo1

HP eh 1047au@TARUN MINGW64 ~/destop/git-demo1 (master)
$ pwd
/c/Users/HP eh 1047au/destop/git-demo1

HP eh 1047au@TARUN MINGW64 ~/destop/git-demo1 (master)
$ ls -al
total 0
drwxr-xr-x 1 HP eh 1047au 197121 0 Jun  2 18:02 ./
drwxr-xr-x 1 HP eh 1047au 197121 0 Jun  2 18:02 ../

HP eh 1047au@TARUN MINGW64 ~/destop/git-demo1 (master)
$ git init jffjfhfhfhfhfhfhjf
Initialized empty Git repository in C:/Users/HP eh 1047au/destop/git-demo1/jffjfhfhfhfhfhfhjf/.git/

HP eh 1047au@TARUN MINGW64 ~/destop/git-demo1 (master)
$ git init
Initialized empty Git repository in C:/Users/HP eh 1047au/destop/git-demo1/.git/

HP eh 1047au@TARUN MINGW64 ~/destop/git-demo1 (master)
$ nano hello.txt

HP eh 1047au@TARUN MINGW64 ~/destop/git-demo1 (master)
$ cat hello.txt
this is a nano filewq
wq
```

2. Three-Stage Workflow Demonstration:

#Commands

1.git status

2.git add hello.txt

3.git status

4.git commit -m "add hello.txt"

5.git status

6.nano hello.txt

7.cat hello.txt

8.git status

9.git diff hello.txt

10.git add hello.txt

11.git status

12.git commit -m "update hello.txt to new file"

#Output

```
HP eh 1047au@TARUN MINGW64 ~/destop/git-demo (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        hello.txt

nothing added to commit but untracked files present (use "git add" to track)

HP eh 1047au@TARUN MINGW64 ~/destop/git-demo (master)
$ git add hello.txt
warning: in the working copy of 'hello.txt', LF will be replaced by CRLF the next time Git touches it

HP eh 1047au@TARUN MINGW64 ~/destop/git-demo (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   hello.txt

HP eh 1047au@TARUN MINGW64 ~/destop/git-demo (master)
$ git commit -m"added hello.txt"
[master (root-commit) 5b80981] added hello.txt
1 file changed, 2 insertions(+)
create mode 100644 hello.txt

HP eh 1047au@TARUN MINGW64 ~/destop/git-demo (master)
$ git status
On branch master

nothing to commit, working tree clean

HP eh 1047au@TARUN MINGW64 ~/destop/git-demo (master)
$ nano hello.txt

HP eh 1047au@TARUN MINGW64 ~/destop/git-demo (master)
$ cat hello.txt
touch hello.txt

Learning git in fun

HP eh 1047au@TARUN MINGW64 ~/destop/git-demo (master)
$
```

```

HP eh 1047au@TARUN MINGW64 ~/destop/git-demo1 (master)
$ git init
Initialized empty Git repository in C:/Users/HP eh 1047au/destop/git-demo1/.git/

HP eh 1047au@TARUN MINGW64 ~/destop/git-demo1 (master)
$ nano hello.txt

HP eh 1047au@TARUN MINGW64 ~/destop/git-demo1 (master)
$ cat hello.txt
this is a nano filewq
wq

HP eh 1047au@TARUN MINGW64 ~/destop/git-demo1 (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        hello.txt
        jfjfhfhfhfhfhfhjf/

nothing added to commit but untracked files present (use "git add" to track)

HP eh 1047au@TARUN MINGW64 ~/destop/git-demo1 (master)
$ git diff hello.txt

HP eh 1047au@TARUN MINGW64 ~/destop/git-demo1 (master)
$ git add hello.txt
warning: in the working copy of 'hello.txt', LF will be replaced by CRLF the next time Git touches it

HP eh 1047au@TARUN MINGW64 ~/destop/git-demo1 (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   hello.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        jfjfhfhfhfhfhfhjf/

HP eh 1047au@TARUN MINGW64 ~/destop/git-demo1 (master)
$ git commit -m "update with a new line"
[master (root-commit) e3cb3fb] update with a new line
1 file changed, 3 insertions(+)
create mode 100644 hello.txt

```

3. Tracking File States:

#Commands

- 1.git status
- 2.nano hello.txt
- 3.cat hello.txt
- 4.git diff hello.txt
- 5.git add hello.txt
- 6.git diff
- 7.git log

8.git commit -m "add third line to hello.txt"

9.git log --oneline

#Output

```
MINGW64:/c/Users/HP eh 1047au/destop/git-demo
On branch master
nothing to commit, working tree clean

HP eh 1047au@TARUN MINGW64 ~/destop/git-demo (master)
$ nano hello.txt

HP eh 1047au@TARUN MINGW64 ~/destop/git-demo (master)
$ cat hello.txt
touch hello.txt

learning git in fun

git has a graet staging area

HP eh 1047au@TARUN MINGW64 ~/destop/git-demo (master)
$ git diff hello.txt
warning: in the working copy of 'hello.txt', LF will be replaced by CRLF the next time Git touches it
diff --git a/hello.txt b/hello.txt
index 4452afe..811774e 100644
--- a/hello.txt
+++ b/hello.txt
@@ -1,3 +1,8 @@
 touch hello.txt

learning git in fun
+
+git has a graet staging area
+
+
HP eh 1047au@TARUN MINGW64 ~/destop/git-demo (master)
$ git add hello.txt
warning: in the working copy of 'hello.txt', LF will be replaced by CRLF the next time Git touches it

HP eh 1047au@TARUN MINGW64 ~/destop/git-demo (master)
$ git diff

HP eh 1047au@TARUN MINGW64 ~/destop/git-demo (master)
$ git log
commit 0b58af06784c02a90d3cfbaf44ee90cec16390db (HEAD -> master)
Author: prathish0169 <prathish.achar@s.amity.edu>
Date: Mon Jun 2 12:18:30 2025 +0530

    update hello.txt with new line

commit 5b809818b182c1b25cc0f2b380d6913d995b2d16
Author: traun28 <tarun.a@s.amity.edu>
Date: Mon Jun 2 12:10:15 2025 +0530

    added hello.txt

HP eh 1047au@TARUN MINGW64 ~/destop/git-demo (master)
$
```

cle

Challenges Faced:

No Challenges faced. Learning Outcomes:

1. Understanding of Git's internal data model with commits, trees, and blobs
2. Practical experience with the three-stage workflow (working directory, staging area, repository)
3. Ability to track and manipulate file states (untracked, modified, staged, unmodified)
4. Experience with Git's diff commands to visualize changes between different stages

Conclusion:

Successfully explored and implemented Git's workflow mechanisms, gaining practical understanding of how Git tracks changes through its staging area. Demonstrated the ability to monitor and manipulate file states throughout the development process, creating a foundation for efficient version control practices.

Lab Exercise 6: Creating and Switching Branches

Objective:

To understand and practice Git's branching system, including creating branches, switching between branches, and managing branch operations.

Prerequisites:

1. Git repository initialization
2. At least one commit in the repository
3. Basic understanding of Git commands (init, add, commit)
4. Command line interface proficiency

Environment Details:

1. Git Version: [2.47.1.windows.2]
2. Working Directory: [/c/user/tej94/mygitproject]

Implementation Steps:

1. Branch Creation:

#Commands

Initialize repository

```
git init
```

```
# Create initial commit  echo "# Git Branching
```

```
Lab" >README.md  git add README.md  git
```

```
commit -m "initial commit:added README.md"
```

```
# Create new branch  git
```

```
branch feature-branch
```

```
# Create and switch to a new branch in one command
```

```
Git checkout -b dev-branch
```

```
#Output
```



```
MINGW64/c/Users/HP eh 1047au/pratathish2
HP eh 1047au@TARUN MINGW64 ~/desktop/git-demo (master)
$ mkdir pratathish2
HP eh 1047au@TARUN MINGW64 ~/desktop/git-demo (master)
$ cd
HP eh 1047au@TARUN MINGW64 ~
$ mkdir pratathish2
HP eh 1047au@TARUN MINGW64 ~
$ cd pratathish2
bash: cd: pratathish2: No such file or directory
HP eh 1047au@TARUN MINGW64 ~
$ cd pratathish2
bash: cd: pratathish2: No such file or directory
HP eh 1047au@TARUN MINGW64 ~
$ cd pratathish2
HP eh 1047au@TARUN MINGW64 ~/pratathish2
$ git init
Initialized empty Git repository in C:/Users/HP eh 1047au/pratathish2/.git/
HP eh 1047au@TARUN MINGW64 ~/pratathish2 (master)
$ echo "# git branching lab" >README.md
$ git add README.md
fatal: pathspec 'README.md' did not match any files
HP eh 1047au@TARUN MINGW64 ~/pratathish2 (master)
$ echo "# git branching lab" >README.md
HP eh 1047au@TARUN MINGW64 ~/pratathish2 (master)
$ git add README.md
warning: in the working copy of 'README.md', LF will be replaced by CRLF the next time Git touches it
HP eh 1047au@TARUN MINGW64 ~/pratathish2 (master)
$ git commit -m "initial commit: added README.md"
[master (root-commit) 48d4ba6] initial commit: added README.md
1 file changed, 1 insertion(+)
create mode 100644 README.md
HP eh 1047au@TARUN MINGW64 ~/pratathish2 (master)
$ git branch feature-branch
HP eh 1047au@TARUN MINGW64 ~/pratathish2 (master)
$ git checkout -b dev-branch
Switched to a new branch 'dev-branch'
HP eh 1047au@TARUN MINGW64 ~/pratathish2 (dev-branch)
$
```

2. Switching Between Branches:

#Commands

Switch back to main branch

git checkout main

View all branches in repository

git branch

Switch to another branch git

checkout feature-branch # Use

newer Git switch command

git switch dev-branch

#Output

```
MINGW64/c/Users/HP eh 1047au/pratathish2
HP eh 1047au@TARUN MINGW64 ~/pratathish2 (dev-branch)
$ git branch
* dev-branch
  feature-branch
  master
HP eh 1047au@TARUN MINGW64 ~/pratathish2 (dev-branch)
$ git checkout feature-branch
Switched to branch 'feature-branch'
HP eh 1047au@TARUN MINGW64 ~/pratathish2 (feature-branch)
$ git switch dev-branch
Switched to branch 'dev-branch'
HP eh 1047au@TARUN MINGW64 ~/pratathish2 (dev-branch)
$
```

3. Branch Management:

#Commands

List all local branches

`git branch`

List all branches (including remote)

`git branch -a`

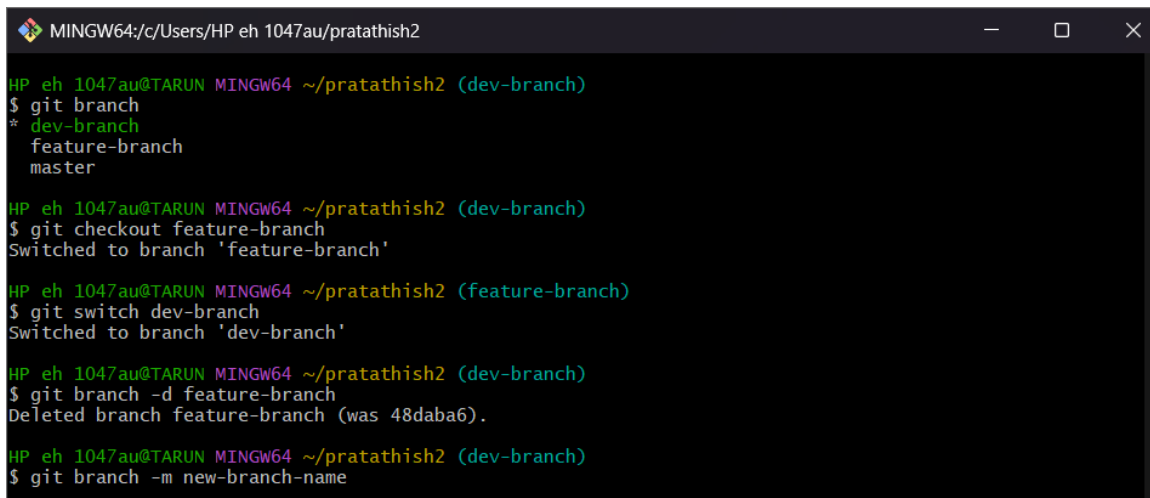
Delete a branch

`git branch -d feature-branch` #

Rename current branch `git`

`branch -m new-branch-name`

#Output



```
MINGW64:/c/Users/HP eh 1047au/pratathish2
HP eh 1047au@TARUN MINGW64 ~/pratathish2 (dev-branch)
$ git branch
* dev-branch
  feature-branch
  master

HP eh 1047au@TARUN MINGW64 ~/pratathish2 (dev-branch)
$ git checkout feature-branch
Switched to branch 'feature-branch'

HP eh 1047au@TARUN MINGW64 ~/pratathish2 (feature-branch)
$ git switch dev-branch
Switched to branch 'dev-branch'

HP eh 1047au@TARUN MINGW64 ~/pratathish2 (dev-branch)
$ git branch -d feature-branch
Deleted branch feature-branch (was 48daba6).

HP eh 1047au@TARUN MINGW64 ~/pratathish2 (dev-branch)
$ git branch -m new-branch-name
```

Challenges Faced:

[Document any issues encountered while creating branches, switching between them, or managing branch operations]

Learning Outcomes:

1. Understanding Git's branching model
2. Ability to create multiple branches for parallel development work
3. Experience switching between different branches to isolate development tasks
4. Practical knowledge of branch management tasks including deletion and renaming

Conclusion:

Successfully explored and implemented Git's branching functionality, gaining practical understanding of how branches enable parallel workflows and feature isolation. Demonstrated the ability to create, navigate between, and manage branches effectively, establishing a foundation for collaborative development practices.

Lab Exercise 07 : Merging branches and resolving conflicts

Objective : To understand and practice two fundamental git merge strategies : Fast – Forward Merge and Three-Way Merge.

Prerequisites :

- Git installation.
- Basic understanding of Git branches. - Command line interface familiarity.

Environment Setup :

1 . Fast-Forward Merge Demonstration :

Scenario : Create a repository and demonstrate a simple, linear merge where no additional changes exist in main branch.

Commands :

```
$ pwd
$ mkdir demo
$ cd demo
$ git init
$ ls -al
$ pwd
$ nano README.md
1234
$ cat README.md
$ git add README.md
$ git commit -m "1234 to README.md"
$ git status
$ git branch
$ git log --oneline
$ git checkout -b b1
$ git branch
$ ls -al
$ cat README.md
$ nano README.md
12345678
$ git add README.md
$ git commit -m "+5678 to README.md"

$ git log --oneline
$ git checkout master
```

```
$ git log --oneline
$ cat README.md
$ git branch
$ git switch master
$ git merge b1
$ cat README.md
$ git branch -d b1
$ git branch
```

Outputs :

```
$ git init
```

```
Initialized empty Git repository in C:/Users/nagas/demo/.git/
```

```
$ git commit -m "1234 to README.md"
```

```
[master (root-commit) e8f6a67] 1234 to README.md
1 file changed, 1 insertion(+)
create mode 100644 README.md
```

```
$ git branch
```

```
* master
```

```
$ git log --oneline
```

```
e8f6a67 (HEAD -> master) 1234 to README.md
```

```
$ git checkout -b b1
```

```
Switched to a new branch 'b1'
```

```
$ git commit -m "+5678 to README.md"
```

```
[b1 4ed8088] +5678 to README.md
1 file changed, 1 insertion(+), 1 deletion(-)
```

```
$ git log --oneline
```

```
4ed8088 (HEAD -> b1) +5678 to README.md
e8f6a67 (master) 1234 to README.md
```

```
$ git merge b1
```

```
Updating e8f6a67..4ed8088
Fast-forward
 README.md | 2 +-
1 file changed, 1 insertion(+), 1 deletion(-)
```

```
$ git branch -d b1
```

```
Deleted branch b1 (was 4ed8088).
```

2 . Three-Way merge Demonstration :

Scenario : Create a scenario where both main and feature branches have different commits.

Commands :

```
$ pwd
$ mkdir demo
$ cd demo
$ git init
$ nano README.md
$ git add README.md
$ git commit -m "1234 to README"
$ git checkout
$ ls -al
$ cat README.md
$ nano README.md
  12345678
$ cat README.md
$ git add README.md
$ git commit -m "+5678 to README"
$ git checkout master
$ cat README.md
$ nano README.md
  1234910
$ cat README.md
$ git add README.md
$ git commit -m "+910 to README"
$ git log --oneline
$ git checkout b1
```

Output :

```
$ git init
```

```
Initialized empty Git repository in C:/Users/nagas/demo/demo/.git/
```

```
$ git commit -m "1234 to README.md"
```

```
[master (root-commit) 0743c44] 1234 to README.md
1 file changed, 2 insertions(+)
create mode 100644 README.md
```

```
$ git checkout -b b1
```

```
Switched to a new branch 'b1'
```

\$ git commit -m "+5678 to README.md"

```
[b1 9f1c5d7] +5678 to README.md  
1 file changed, 1 insertion(+), 1 deletion(-)
```

\$ git checkout master

```
Switched to branch 'master'
```

\$ git commit -m "+910 to README.md"

```
[master 3f30e0a] +910 to README.md  
1 file changed, 1 insertion(+), 1 deletion(-)
```

Potential Merge Conflict :

Commands :

\$ git log --oneline

\$ git branch

\$ git checkout master

\$ git merge b1

Output :

\$ git merge b1

```
Auto-merging README.md  
CONFLICT (content): Merge conflict in README.md  
Automatic merge failed; fix conflicts and then commit the result.
```

Resolving Conflicts :

Commands :

\$ nano README.md

```
<<<<HEAD
```

```
1234910
```

```
-----
```

```
-----
```

```
12345678
```

```
>>>>b1
```

\$ cat README.md

\$ git add README.md

\$ git commit -m "resolve conflict : 12345678910 to README.md"

\$ git branch

\$ git merge b1

\$ git branch -d b1

```
$ git branch
$ git log --oneline
```

```
# Output :
$ cat README.md
```

```
<<<<<<< HEAD
1234910
=====
12345678
>>>>>>> b1
```

```
$ git commit -m "resolve conflict : 12345678910 to README.md"
```

```
[master 60403cc] resolve conflict : 12345678910 to README.md
```

```
$ git log --oneline
```

```
60403cc (HEAD -> master) resolve conflict : 12345678910 to README.md
3f30e0a +910 to README.md
9f1c5d7 +5678 to README.md
0743c44 1234 to README.md
```

Learning Outcomes:

- 1.Understanding different Git merge strategies.
- 2.Practicing branch merging techniques.
- 3.Recognizing when different merge approaches are used.

Conclusion :

Merging is a fundamental Git operation that allows developers to combine work from different branches, enabling parallel development and collaborative workflows.

Lab Exercise 08 : Working with Remote Repositories.

Objective : To understand and practice working with remote Git repositories, including connecting to remote repositories, pushing local changes, and pulling remote changes.

Prerequisites :

- Git installed
- Basic understanding of Git branches and commits
- Command line interface familiarity
- GitHub/GitLab/Bitbucket account (or any Git hosting service)

Environment Setup :

Case 1: Pushing a Local Repository to a Remote Repository

Scenario: You have a local project that you want to share by pushing it to a remote repository.

Commands :

```
$ pwd
$ cd Desktop $
mkdir ex08
$ cd ex08 $
git init
$ pwd $
ls -al
$ echo "WIP on a decided stuff.." > README.md
$ cat README.md
$ git add README.md
$ git commit -m "Initial Commit"
$ git status
$ git log --oneline $ git
remote add origin
https://github.com/00PoojashreeK/exercise8.git
$ git remote -v
$ git branch
$ git status
$ git log --oneline
$ git push -u origin main
```

Output :

\$ git init

```
Initialized empty Git repository in C:/Users/nagas/ex08/.git/
```

\$ cat README.md

```
WIP on a decided stuff..
```

\$ git commit -m "Initial Commit"

```
[master (root-commit) 64a0d0c] Initial commit
1 file changed, 1 insertion(+)
create mode 100644 README.md
```

\$ git log --oneline

```
64a0d0c (HEAD -> master) Initial commit
```

\$ git remote -v

```
sukruth gowda@chukki MINGW64 ~ (dev-branch)
$ ^[[200~origin https://github.com/sukruthgowda/exercise8.git (fetch)
bash: syntax error near unexpected token `('

sukruth gowda@chukki MINGW64 ~ (dev-branch)
$ origin https://github.com/sukruthgowda/exercise8.git (push)
bash: syntax error near unexpected token `('
```

Case 2 : Cloning and Pulling from a Remote Repository

Scenario : A remote repository exists that you want to work with locally. You'll clone it, observe changes made to the remote repository, and pull those changes to your local copy.

Commands :

\$ cd Desktop

\$ mkdir demo

\$ cd demo

\$ pwd \$

ls -al

\$ git clone <https://github.com/sukruthgowda/exercise8.git>

Output :

\$ git clone <https://github.com/sukruthgowda/exercise8.git>

```
$ git clone https://github.com/00PoojashreeK/exercise8.git
Cloning into 'exercise8'...
remote: Enumerating objects: 6, done.
remote: Counting objects: 100% (6/6), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 6 (delta 0), reused 6 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (6/6), done.
```

Handling Authentication and Potential issues

Commands :

```
$ git remove -v
$ pwd
$ git pull origin main
```

Output :

```
$ git pull origin main
```

```
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 3 (delta 0), pack-reused 0 (from 0)
Unpacking objects: 100% (3/3), 219 bytes | 4.00 KiB/s, done.
From https://github.com/00PoojashreeK/exercise8
* branch          main      -> FETCH_HEAD
* [new branch]     main      -> origin/main
fatal: refusing to merge unrelated histories
```

Learning Outcomes :

- 1 . Understanding how to connect local repositories to remote repositories.
- 2 . Learning to push local changes to remote repositories
- 3 . Learning to pull remote changes to local repositories
- 4 . Managing typical remote repository workflows

Conclusion :

Working with remote repositories is essential for collaborative software development. It allows teams to share code, track changes, and maintain project history across multiple developers and workstations.

Lab exercise 09 : Collaborating using Pull Requests

Objective :

To understand and practice the GitHub pull request workflow , including creating branches, making changes , submitting pull requests, reviewing code, and merging changes.

Prerequisites :

- GitHub account
- Basic understanding of Git concepts
- Web browser

Environment :

This lab was completed entirely through the GitHub web interface, no local Git installation was required.

Case 1 : Create a New Repository and Branch

Scenario : Set up a new project repository and create a feature branch for development.

Commands Used

No local commands needed – used GitHub web interface

Steps Completed and Output 1

. Created a new repository.

- Navigated to GitHub.com and signed in
- Clicked the “+” icon in the top-right corner
- Selected “New repository”
- Named it: “pull-request-practice”
- Description: “Repository for practicing pull requests”
- Visibility: Public
- Checked “Add a README file”

Output :

\$ Created a new repository

Start a new repository for 00PoojashreeK

A repository contains all of your project's files, revision history, and collaborator discussion.

Repository name *

test

⚠ The repository test already exists on this account.

☒ Public

Anyone on the internet can see this repository

☐ Private

You choose who can see and commit to this repository

Create a new repository

Introduce yourself with a profile README

Share information about yourself by creating a profile README, which appears at the top of your profile page.

00PoojashreeK / README.md

Create

\$ Checked README.md

main

1 Branch

0 Tags

Go to file

t

+

<> Code

00PoojashreeK

Merge pull request #1 from 00PoojashreeK/feature-branch

1231bd5 · last week

3 Commits

README.md

Update README.md

last week

README

test

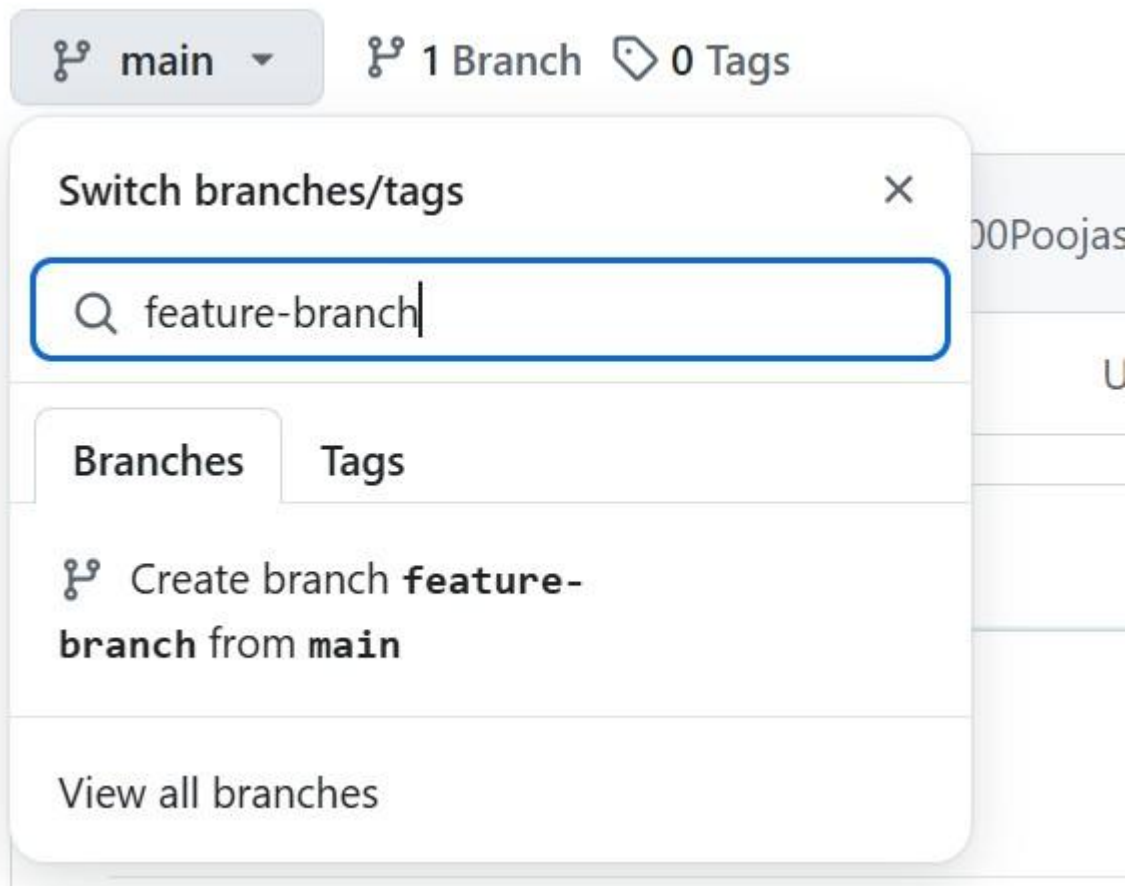
##test123

2 . Created a new branch:

- In the repository , clicked the “main” branch dropdown
- Typed “feature-branch” in the search box
- Clicked “Create branch: feature-branch from main”

Output:

\$ Checked main branch and created feature-branch



Case 2 : Made changes and created a pull request

Scenario : Made code changes in my feature branch and created a pull request to propose merging these changes to main branch

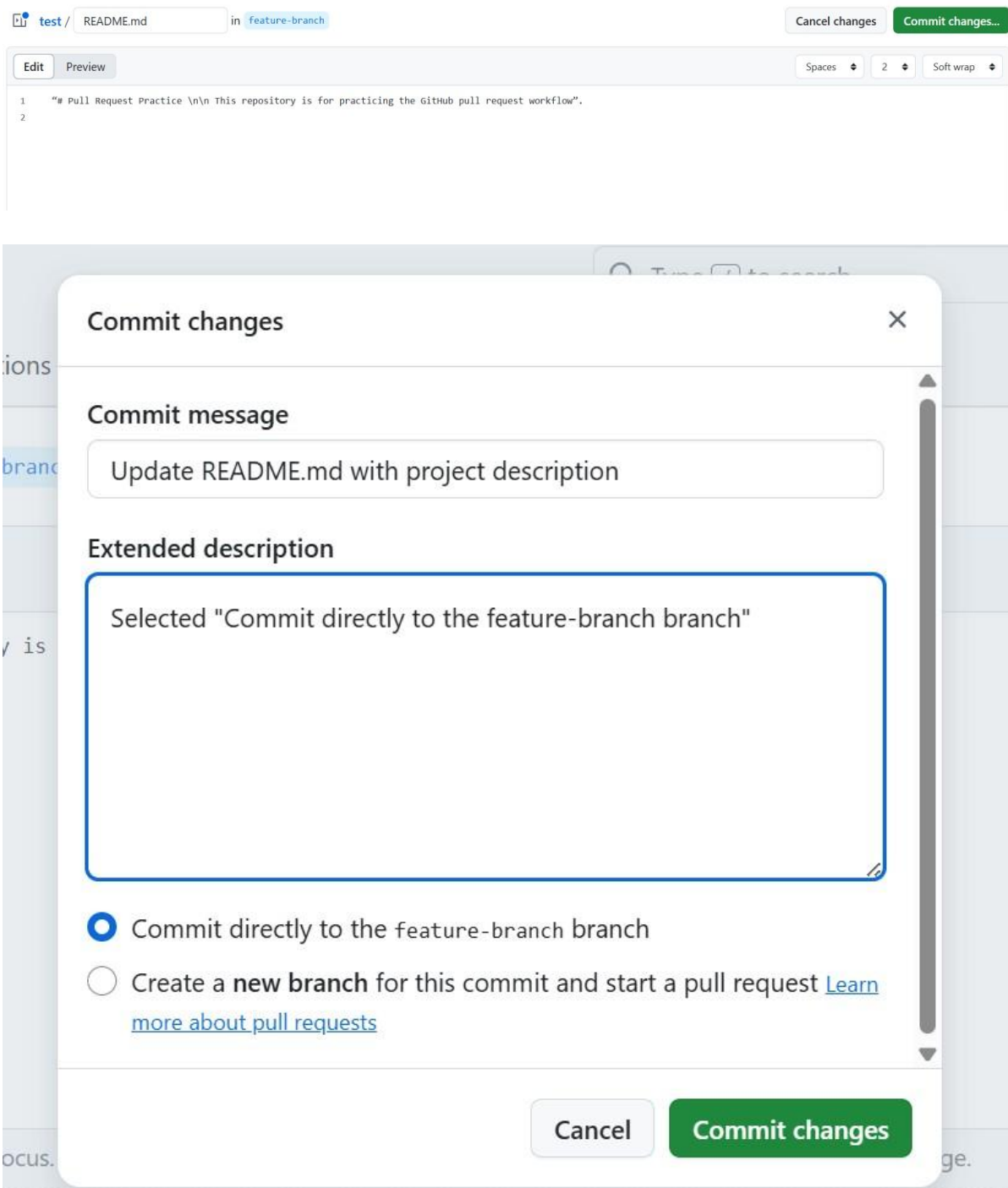
Steps Completed and Output

1 . Edited a file:

- Navigated to the README.md file
- Clicked the pencil icon to edit
- Added content: “# Pull Request Practice \n\n This repository is for practicing the GitHub pull request workflow”.
- Scrolled down and added commit message: Update README with project description”
- Selected “Commit directly to feature-branch branch”
- Clicked “Commit changes”

Output:

\$ Added a content to README.md



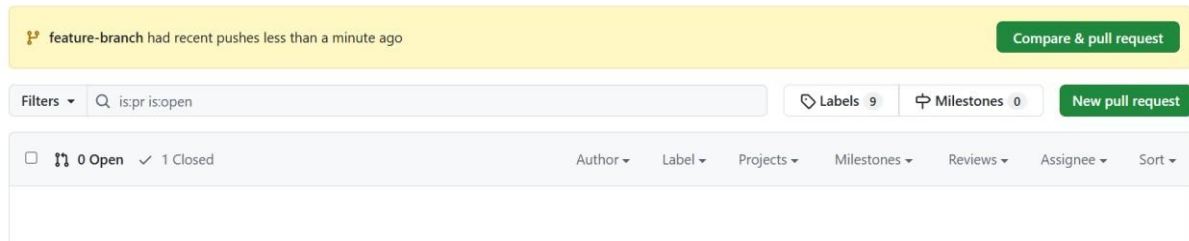
2 . Created a pull request:

- Clicked on "Pull requests" tab
- Clicked "New pull request"
- Set "base: main" and "compare: feature-branch"
- Reviewed the changes
- Clicked "Create pull request"
- Title: "Add project description to README"

- Description: "Updated the README file with information about the repository purpose."
- Clicked "Create pull request"

Output:

\$ clicked on pull request

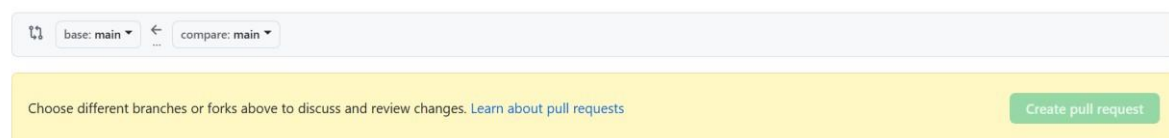


A screenshot of the GitHub pull request interface. At the top, a yellow banner shows a notification for 'feature-branch' with recent pushes. Below this is a search bar with 'is:pr is:open' and filters for 'Labels' (9) and 'Milestones' (0). A table header shows '0 Open' and '1 Closed' pull requests, with columns for Author, Label, Projects, Milestones, Reviews, Assignee, and Sort.

\$ clicked new pull request

Compare changes

Compare changes across branches, commits, tags, and more below. If you need to, you can also [compare across forks](#).



A screenshot of the 'Compare changes' interface. It shows a comparison between 'base: main' and 'compare: main'. Below this is a yellow banner with the text 'Choose different branches or forks above to discuss and review changes. Learn about pull requests' and a 'Create pull request' button.

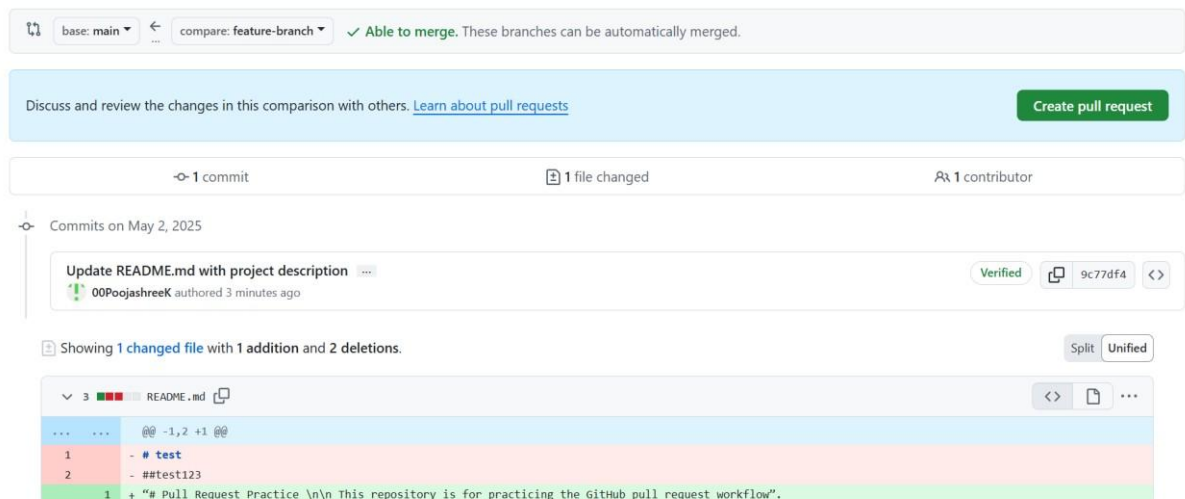


Compare and review just about anything

Branches, tags, commit ranges, and time ranges. In the same repository and across forks.

| Example comparisons | |
|---------------------|--------------|
| feature-branch | 1 minute ago |
| main@{1day}...main | 24 hours ago |

\$ Reviewed the changes



A screenshot of the GitHub pull request details page. At the top, it shows a comparison between 'base: main' and 'compare: feature-branch' with a status 'Able to merge'. Below this is a blue banner with the text 'Discuss and review the changes in this comparison with others. Learn about pull requests' and a 'Create pull request' button. The main section shows '1 commit' and '1 file changed'. A commit titled 'Update README.md with project description' by '00PoojashreeK' is shown. Below the commit, it says 'Showing 1 changed file with 1 addition and 2 deletions.' and shows a diff for 'README.md'.

\$ clicked create pull request



Add a title

Update README.md with project description

Add a description

Write

Preview

H B I [list icon] [code icon] [link icon] [table icon] [table icon] [table icon] [image icon] [mention icon] [share icon] [undo icon] [redo icon]

Selected "Commit directly to the feature-branch branch"

Markdown is supported

Paste, drop, or click to add files

\$ Added title and description



Add a title

Add project description to README

Add a description

Write

Preview

H B I [list icon] [code icon] [link icon] [table icon] [table icon] [table icon] [image icon] [mention icon] [share icon] [undo icon] [redo icon]

Updated the README file with information about the repository purpose

Markdown is supported

Paste, drop, or click to add files

Create pull request

Remember, contributions to this repository should follow our [GitHub Community Guidelines](#)

\$ clicked on create pull request

Add project description to README #2

Edit <> Code

Open 00PoojashreeK wants to merge 1 commit into main from feature-branch

Conversation 0 Commits 1 Checks 0 Files changed 1 +1 -2

00PoojashreeK commented now

Updated the README file with information about the repository purpose

Update README.md with project description

Verified 9c77df4

No conflicts with base branch
Merging can be performed automatically.

Merge pull request

You can also merge this with the command line. [View command line instructions](#)

Reviewers

No reviews

Still in progress? [Convert to draft](#)

Assignees

No one—[assign yourself](#)

Labels

None yet

Projects

None yet

Case 3: Completed the Code Review Process

Scenario: My pull request needed to be reviewed before merging.

Steps Completed and Output 1

. Requested reviewers:

- In the pull request, clicked "Reviewers" on the right sidebar
- Searched for and selected teammates (since this was a solo project, I noted that in a team setting I would add team members here)

Output:

Reviewers

TanusReddy

Still in progress? [Convert to draft](#)

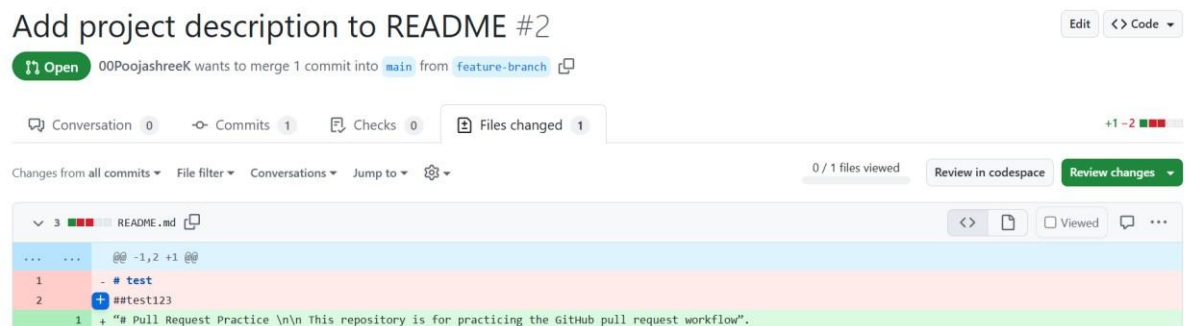
2 . Added a review comment (simulated the reviewer role):

- In the "Files changed" tab, hovered over a line
- Clicked the "+" that appeared
- Added comment: "Consider adding more details about what specific PR workflows we'll practice"
- Clicked "Start a review"
- Clicked "Finish review"

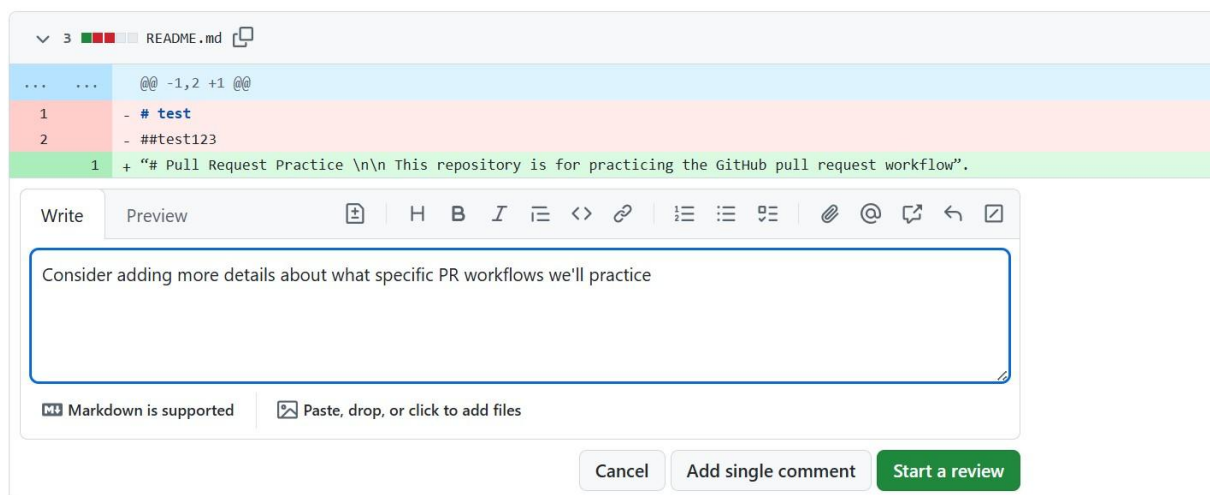
- Selected "Request changes"
- Clicked "Submit review"

Output:

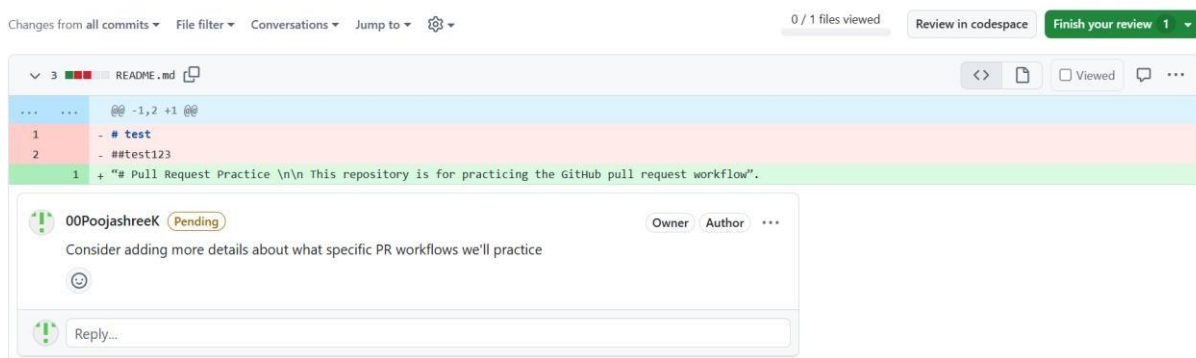
\$ Files changed tab



\$ clicked "+" icon and added a comment



\$ clicked "start a review"



\$ Clicked "finish review" and selected request changes and then clicked submit review

00PoojashreeK commented 2 days ago

Updated the README file with information about the repository purpose

Update README.md with project description

00PoojashreeK requested a review from TanusReddy 2 days ago

00PoojashreeK commented 2 minutes ago

View reviewed changes

README.md

```

... @@ -1,2 +1 @@
1 - # test
2 - ##test123
1 + "# Pull Request Practice \n\n This repository is for practicing the GitHub pull request workflow".

```

00PoojashreeK 2 days ago

Consider adding more details about what specific PR workflows we'll practice

Reply...

Resolve conversation

Reviewers

TanusReddy

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.

Notifications

Unsubscribe

You're receiving notifications because you're watching this repository.

1 participant

Lock conversation

Review requested

Review has been requested on this pull request. It is not required to merge. [Learn more about requesting a pull request review.](#)

1 pending review

No conflicts with base branch

Merging can be performed automatically.

Merge pull request

You can also merge this with the command line. [View command line instructions.](#)

Add a comment

Write Preview

H B I

3 . Addressed review comments:

- Went back to the README.md file and clicked the pencil icon
- Added: "We will practice creating branches, making changes, creating pull requests, reviewing code, and merging changes."
- Commit message: "Add more details as requested in the review"
- Clicked "Commit changes"

Output:

\$ Added one line in README.md file

test / README.md in main

Cancel changes Commit changes...

Edit Preview

Spaces 2 Soft wrap

```

1 # test
2 ##test123
3 We will practice creating branches , making changes , creating pull requests, reviewing code,and merging changes.

```

\$ Clicked on commit changes

Commit changes

Commit message

Update README.md

Extended description

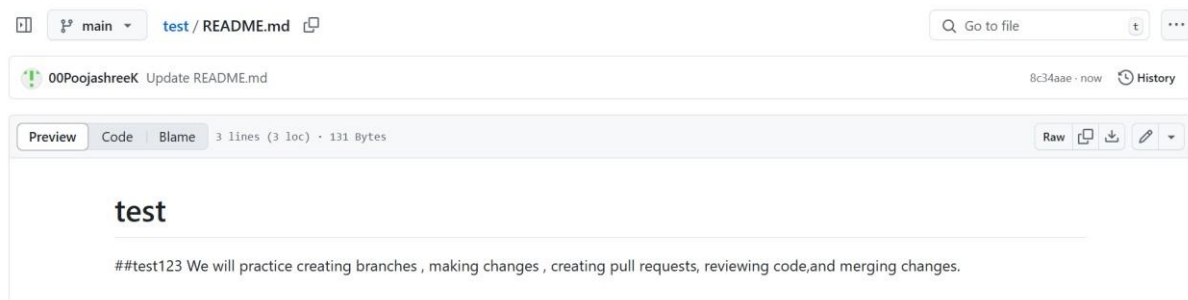
Add an optional extended description...

☒ 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

\$ After clicking commit changes

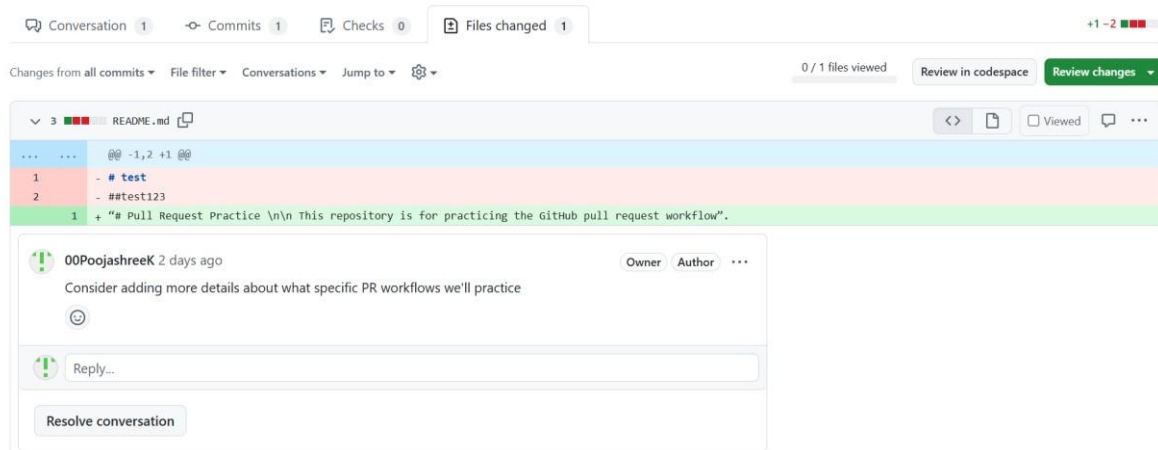


4 . Approved the changes (simulated the reviewer role again):

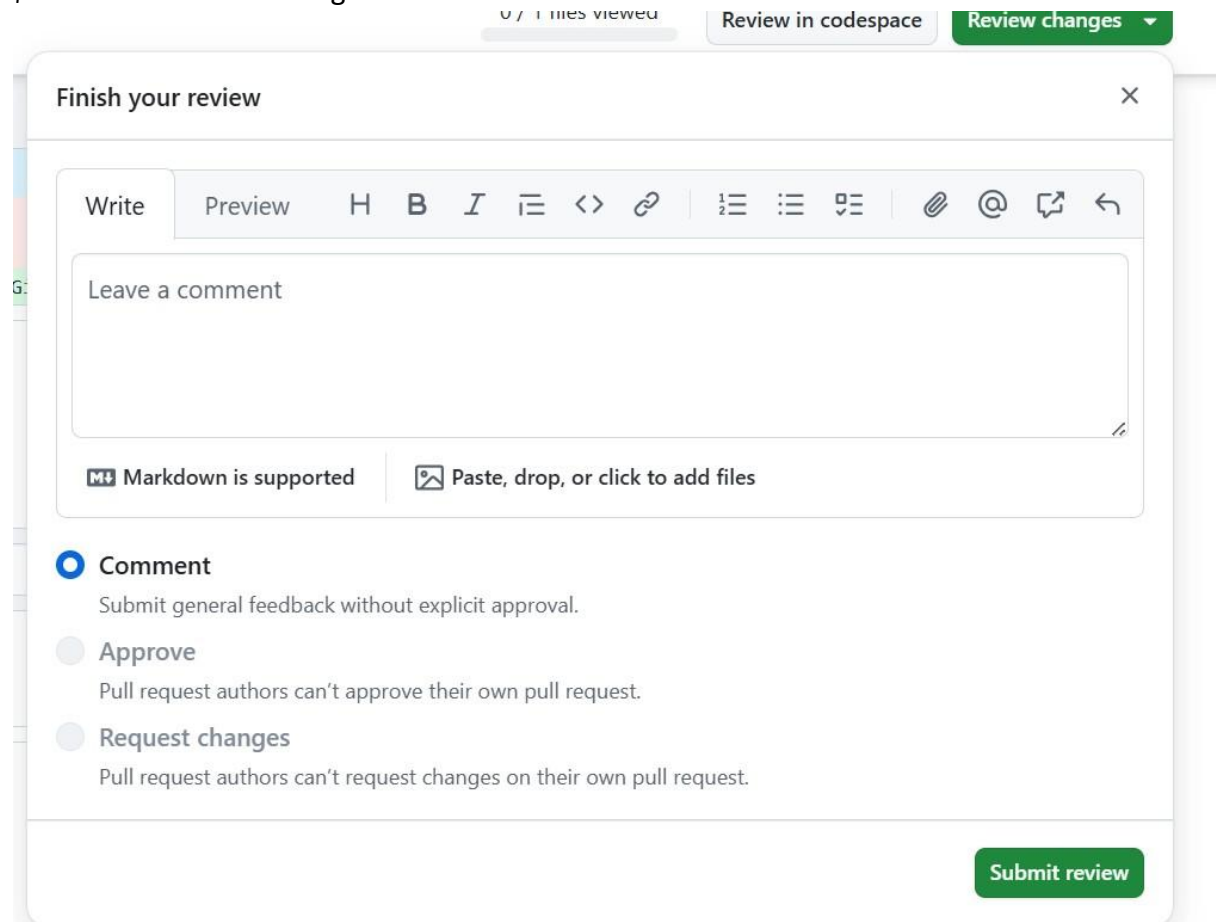
- Went to "Files changed" tab
- Clicked "Review changes"
- Selected "Approve"
- Added comment: "Changes look good now, ready to merge."
- Clicked "Submit review"

Output :

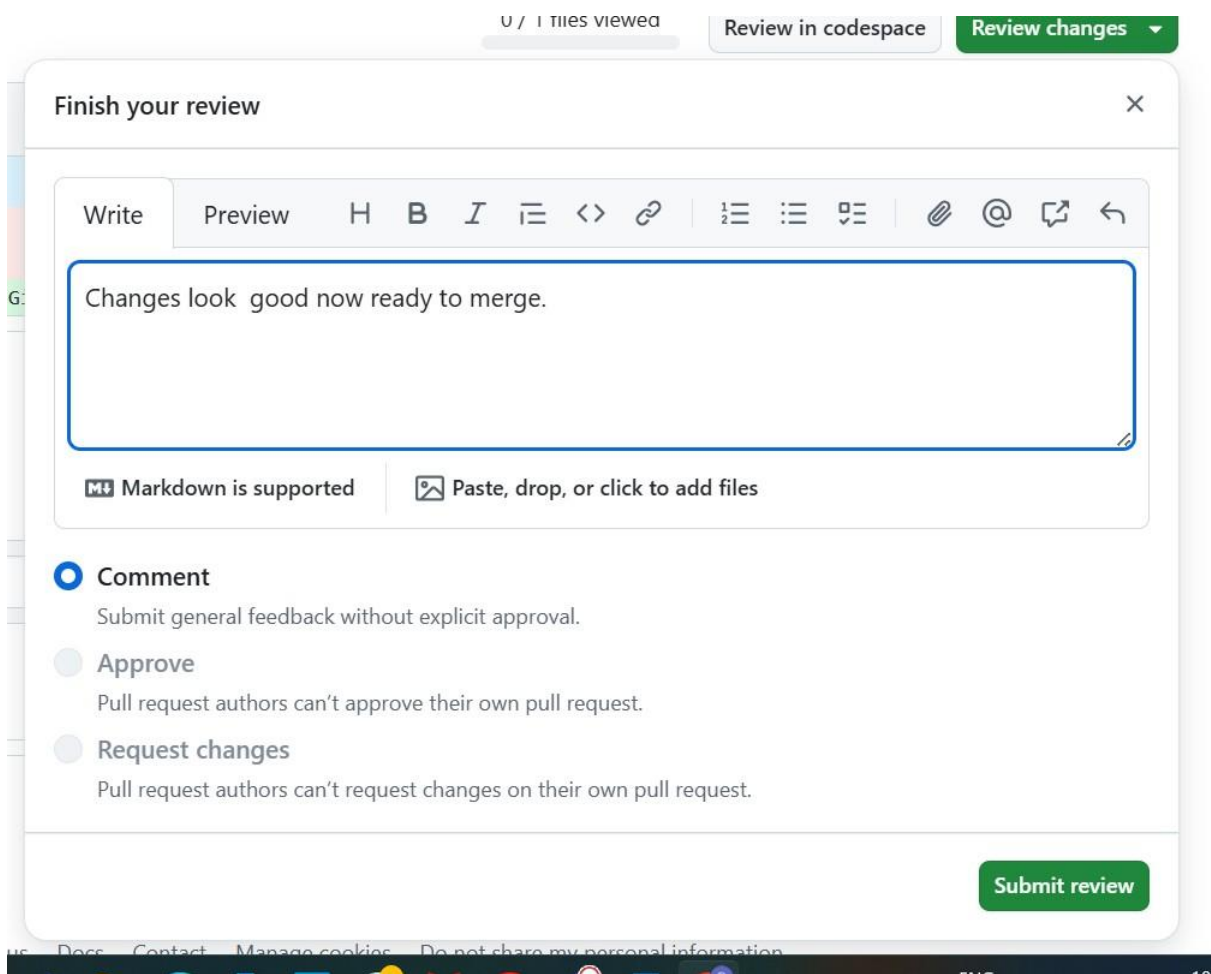
\$ Went back to files changed tab



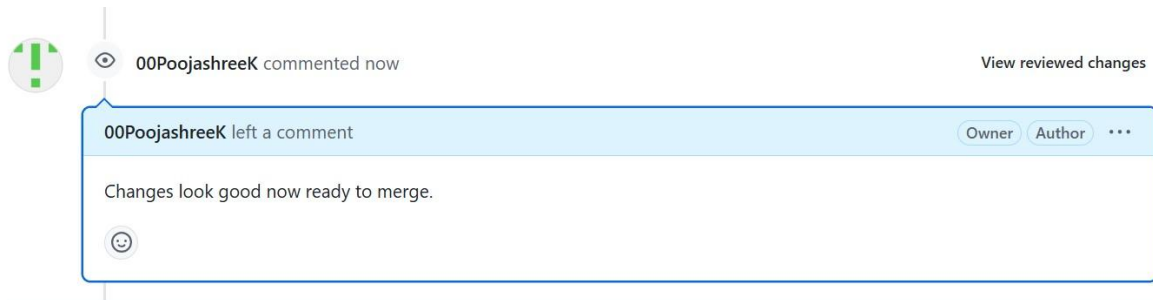
\$ Clicked on review changes



\$ Click on approve and add a comment



\$ Clicked submit review



Case 4 : Merged Changes

Scenario: The pull request was approved and ready to merge.

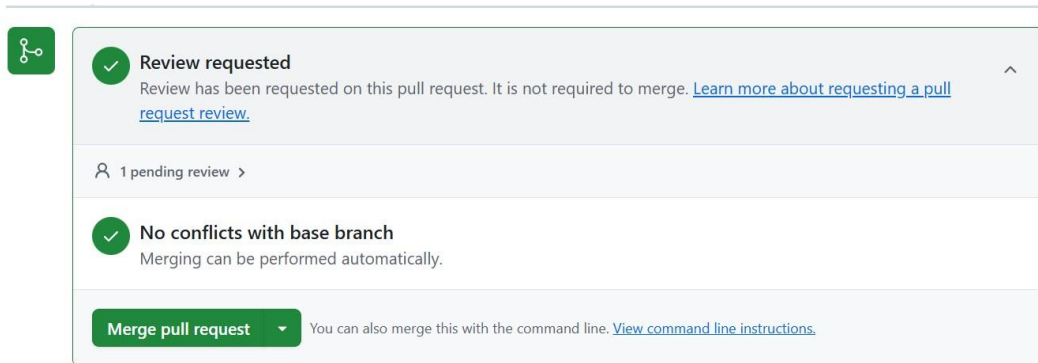
Steps Completed and Output 1

. Merged the pull request:

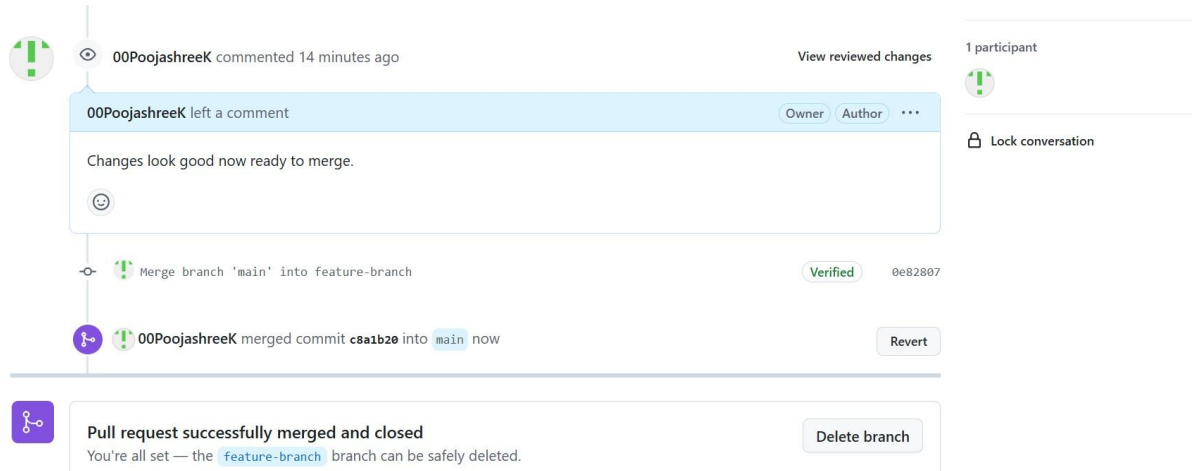
- Went to the "Conversation" tab
- Clicked "Merge pull request"
- Confirmed with "Confirm merge"

Output :

\$ Went back to conversation tab



\$ Clicked on merge pull request and then clicked on confirm merge

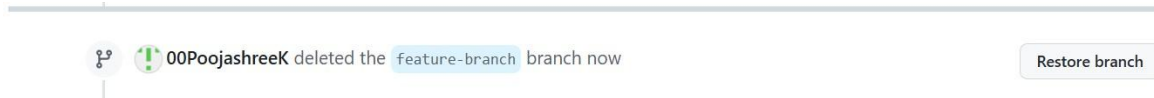


2 . Deleted the branch:

- Clicked "Delete branch" after merge

Output :

\$ Click on delete branch



Handled Potential Issues

Common Issue 1: Merge Conflicts

- Problem: No merge conflicts occurred during my exercise
- Knowledge Gained: I learned how to handle merge conflicts if they had occurred: 1 . Click "Resolve conflicts" button that appears 2 . Edit the file to resolve conflicts (remove conflict markers and keep desired changes) 3 . Click "Mark as resolved" 4 . Click "Commit merge"

Reviewers vs. Assignees - Understanding Gained

- Reviewers: People responsible for examining code changes and providing feedback - I learned that reviewers should be added when technical feedback on implementation is needed
- Reviewers can approve or request changes
- Assignees: People responsible for the overall progress of the pull request
- Usually the person who created the PR or who is currently working on it
- Helps track ownership and responsibility for the pull request

Learning Outcomes Achieved

- 1 . Understood how to create and manage branches on GitHub
- 2 . Learned to create pull requests to propose changes
- 3 . Understood the code review process
- 4 . Learned how to address feedback and merge changes
- 5 . Understood the difference between reviewers and assignees

Conclusion :

I successfully completed the pull request workflow exercise. This process is fundamental to collaborative software development. It provides a structured way to propose, review, discuss, and merge changes, which improves code quality and team coordination. I am now comfortable with the GitHub pull request workflow and ready to apply these skills in team development environments.

