DAY – 3 DEMO 6 : CREATING , SWITCHING AND MERGING BETWEEN PROJECTS]

REFERENCE: HTTPS://WWW.ATLASSIAN.COM/GIT/TUTORIALS/SETTING-UP-A-REPOSITORY/GIT-INIT

Getting started:

- **git init:**

  The git init command creates a new Git repository. It can be used to convert an existing, unversioned project to a Git repository or initialize a new, empty repository

  Executing git init creates a .git subdirectory in the current working directory, which contains all of the necessary Git metadata for the new repository.

git ignore :

Git sees every file in your working copy as one of three things:

1. tracked - a file which has been previously staged or committed;
2. untracked - a file which has not been staged or committed; or
3. ignored - a file which Git has been explicitly told to ignore.

Ignored files are tracked in a special file named .gitignore that is checked in at the root of your repository. There is no explicit git ignore command: instead the .gitignore file must be edited and committed by hand when you have new files that you wish to ignore.

BASIC COMMANDS

- **git clone:** git clone is a Git command line utility which is used to target an existing repository and create a clone, or copy of the target repository on your local.

Usage: git clone <repo> <directory>

```
git clone ssh/http://john@example.com/path/to/my-project.git
cd my-project
```

- **git add :**

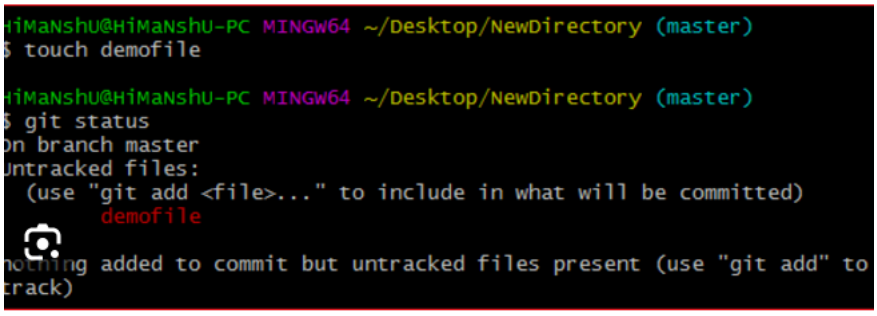
  The git add command adds a change in the working directory to the staging area. It tells Git that you want to include updates to a particular file in the next commit. However, git add doesn't really affect the repository in any significant way—changes are not actually recorded until you run git commit.

  Usage: git add . (Stages all files)
  git add {file_name} - single file

- **git status: shows all the staged/unstaged files in branch**

  The git status command displays the state of the working directory and the staging area. It lets you see which changes have been staged, which haven't, and which files aren't being tracked by Git.

  Usage: git status



- **git commit :**

  The git commit command is one of the core primary functions of Git. Prior use of the git add command is required to select the changes that will be staged for the next commit. Then git commit is used to create a snapshot of the staged changes along a timeline of a Git projects history.

  Usage: git commit -m "{descrtiption of commit }"

- **git rm :**

  The git rm command can be used to remove individual files or a collection of files. The primary function of git rm is to remove tracked files from the Git index. Additionally, git rm can be used to remove files from both the staging index and the working directory
  Usage: git rm Documentation*.txt, git rm -f git-*.sh

Syncing (git remote)

- **git fetch :** The git fetch command downloads commits, files, and refs from a remote repository into your local repo. Fetching is what you do when you want to see what everybody else has been working on**.**

  Usage:

  git fetch <remote>

  git fetch <remote> <branch>

  git fetch all

- **git pull:** The git pull command first runs git fetch which downloads content from the specified remote repository. Then a git merge is executed to merge the remote content refs and heads into a new local merge commit

  Usage: git pull <remote>

Diff Between Pull & Fetch

Git Pull= Git Fetch + Git Merge **-** Git Pull Only bring the remote files to Local but does not merge with the Local Repo

Git Pull downloads the content and merges with the Local Repository

BRANCHING COMMANDS

- **git branch - lists all branches**

  Usage: $> git branch

  main

  another_branch

  feature_inprogress_branch

- **git branch -b {branch_name} : creates a new branch**

- **git checkout {branch_name}: checks/moves to new branch created**

  The git checkout command lets you navigate between the branches created by git branch
  Usage : git checkout master
  Git checkout feature

  Usage: $> git branch

main

another_branch

feature_inprogress_branch

$> git checkout feature_inprogress_branch

- **git checkout -b {branch_name}: creates and checks out to new branch created**

  Usage: git checkout -b <remotebranch>
  Example : git checkout -b feature1

- **git merge {branch_name} : merges the dev/new branch created with master. Has to been only after checking out to master**
  - git merge is used to combine two branches**.**
  - Git merge will combine multiple sequences of commits into one unified history.

  Usage:

  - git checkout master

    git merge feature

- **git log : Lists all the commit id's in the branch**
-

```
HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/mewproject (main)
$ git log
commit ee9159709034dd2acc03ead8add1652dc3e31ca4 (HEAD -> main, origin/main)
Author: bhargavdevops2024 <bhargavdevops2024@gmail.com>
Date:   Tue Nov 26 14:34:43 2024 +0530

    hello file

commit a32879187d073e2503d60c89eff5c62c99fb013a
Author: bhargavdevops2024 <bhargavdevops2024@gmail.com>
Date:   Mon Nov 25 22:27:55 2024 +0530

    Revert "new file"

    This reverts commit 7ae8bf42766816568f9ff3d7452a10d2c1ead503.

commit 7ae8bf42766816568f9ff3d7452a10d2c1ead503
Author: bhargavdevops2024 <bhargavdevops2024@gmail.com>
Date:   Mon Nov 25 22:27:33 2024 +0530

    new file

commit fe1c4c7d8f502c3683c58bde81f4c4501086b722
Author: bhargavdevops2024 <bhargavdevops2024@gmail.com>
Date:   Mon Nov 25 22:00:56 2024 +0530

    new file
```

- **git remote -v:** gives the target repositories to pull and push

```
HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/mewproject (main)
$ git remote -v
origin  https://github.com/bhargavdevops2024/CI-CD-Project-Lab2.git (fetch)
origin  https://github.com/bhargavdevops2024/CI-CD-Project-Lab2.git (push)
```

- **git remote add origin {link to the repositiory} : points git to the repository where files need to be pushed**

  Usage:

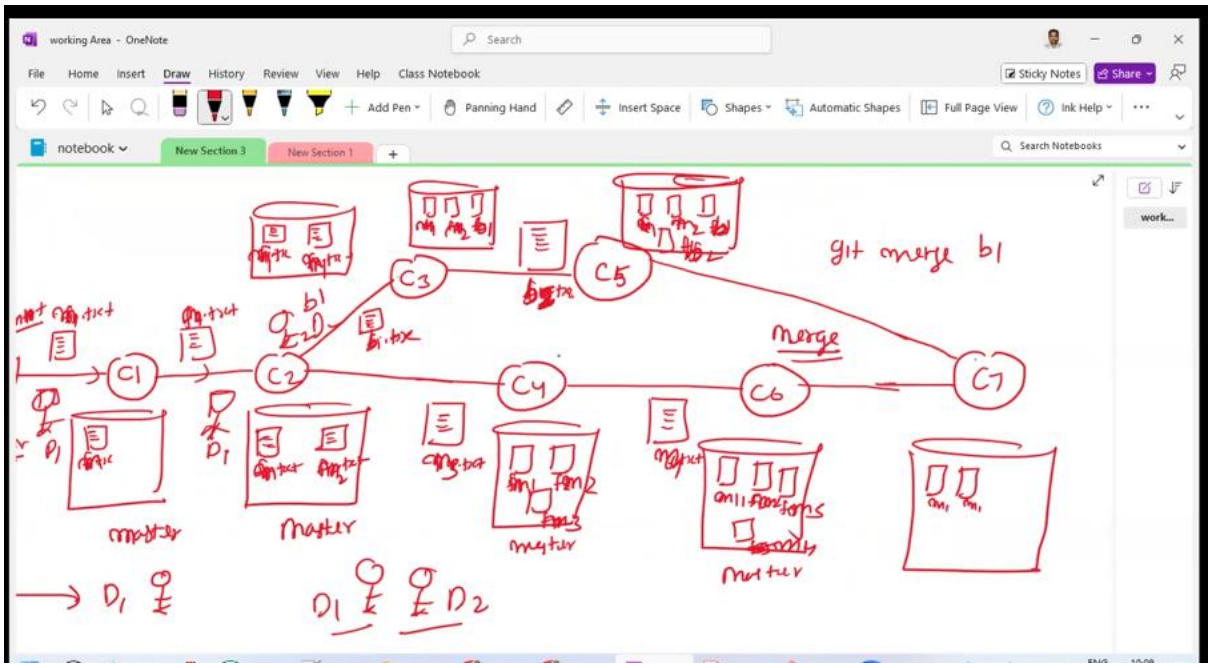
  $ git remote add origin https://github.com/bhargavdevops2024/CI-CD-Project-Lab2.git

- **git push origin {branch_name} : pushes the files to the branch specified**

  Usage:

```
HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/mewproject (main)
$ git push origin main
Everything up-to-date
```

BRANCHING STRATEGY



```
HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/demo-day2/lesson-end-project-lab2 (main)
$ git branch
* main

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/demo-day2/lesson-end-project-lab2 (main)
$ git branch dev

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/demo-day2/lesson-end-project-lab2 (main)
$ git checkout dev
Switched to branch 'dev'

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/demo-day2/lesson-end-project-lab2 (dev)
$ vi index.html

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/demo-day2/lesson-end-project-lab2 (dev)
$ vi index.txt

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/demo-day2/lesson-end-project-lab2 (dev)
$ git status
On branch dev
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        index.txt

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

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/demo-day2/lesson-end-project-lab2 (dev)
$ git add .
warning: in the working copy of 'index.txt', LF will be replaced by CRLF the next time Git touches it

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/demo-day2/lesson-end-project-lab2 (dev)
$ git status
On branch dev
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   index.txt


HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/demo-day2/lesson-end-project-lab2 (dev)
$ git log
commit 09626f08eb8f0e37bc574f6bac5fe4fb866410aa (HEAD -> dev, origin/main, origin/HEAD, main)
Author: bhargavdevops2024 <bhargavdevops2024@gmail.com>
Date:   Tue Nov 26 21:08:04 2024 +0530

    Create m1.txt
```

```
$ git log
commit 09626f08eb8f0e37bc574f6bac5fe4fb866410aa (HEAD -> dev, origin/main, origin/HEAD, main)
Author: bhargavdevops2024 <bhargavdevops2024@gmail.com>
Date:   Tue Nov 26 21:08:04 2024 +0530

    Create m1.txt

commit 3d38733f9311c3045034d8600c47ee53797c15fd
Author: bhargavdevops2024 <bhargavdevops2024@gmail.com>
Date:   Tue Nov 26 21:03:53 2024 +0530

    Initial commit

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/demo-day2/lesson-end-project-lab2 (dev)
$ git checkout master
error: pathspec 'master' did not match any file(s) known to git

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/demo-day2/lesson-end-project-lab2 (dev)
$ git checkout main
Switched to branch 'main'
A       index.txt
Your branch is up to date with 'origin/main'.

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/demo-day2/lesson-end-project-lab2 (main)
$ git log
commit 09626f08eb8f0e37bc574f6bac5fe4fb866410aa (HEAD -> main, origin/main, origin/HEAD, dev)
Author: bhargavdevops2024 <bhargavdevops2024@gmail.com>
Date:   Tue Nov 26 21:08:04 2024 +0530

    Create m1.txt

commit 3d38733f9311c3045034d8600c47ee53797c15fd
Author: bhargavdevops2024 <bhargavdevops2024@gmail.com>
Date:   Tue Nov 26 21:03:53 2024 +0530

    Initial commit

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/demo-day2/lesson-end-project-lab2 (main)
$ ls
README.md  index.txt  m1.txt

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/demo-day2/lesson-end-project-lab2 (main)
$ git checkout dev
Switched to branch 'dev'
A       index.txt
```

```
HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/demo-day2/lesson-end-project-lab2 (main)
$ ls
README.md  index.txt  m1.txt

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/demo-day2/lesson-end-project-lab2 (main)
$ git checkout dev
Switched to branch 'dev'
A       index.txt

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/demo-day2/lesson-end-project-lab2 (dev)
$ ls
README.md  index.txt  m1.txt

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/demo-day2/lesson-end-project-lab2 (dev)
$ git checkout main
Switched to branch 'main'
A       index.txt
Your branch is up to date with 'origin/main'.

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/demo-day2/lesson-end-project-lab2 (main)
$ ls
README.md  index.txt  m1.txt
l
HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/demo-day2/lesson-end-project-lab2 (main)
$ ls -ltr
total 3
-rw-r--r-- 1 HP 197121  5 Nov 26 21:10 m1.txt
-rw-r--r-- 1 HP 197121 25 Nov 26 21:10 README.md
-rw-r--r-- 1 HP 197121  6 Nov 26 21:25 index.txt

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/demo-day2/lesson-end-project-lab2 (main)
$ ls
README.md  index.txt  m1.txt

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/demo-day2/lesson-end-project-lab2 (main)
$ git checkout dev
Switched to branch 'dev'
A       index.txt

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/demo-day2/lesson-end-project-lab2 (dev)
$ ls -ltr
total 3
-rw-r--r-- 1 HP 197121  5 Nov 26 21:10 m1.txt
-rw-r--r-- 1 HP 197121 25 Nov 26 21:10 README.md
-rw-r--r-- 1 HP 197121  6 Nov 26 21:25 index.txt
```

Switch to new branch create c4,c5,c6 files and stage it using git add . , commit and check number of commits using git log

```
HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/demo-day2/lesson-end-project-lab2 (dev)
$ touch c4
touch
HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/demo-day2/lesson-end-project-lab2 (dev)
$ touch c5

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/demo-day2/lesson-end-project-lab2 (dev)
$ touch c6

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/demo-day2/lesson-end-project-lab2 (dev)
$ git log
commit 09626f08eb8f0e37bc574f6bac5fe4fb866410aa (HEAD -> dev, origin/main, origin/HEAD, main)
Author: bhargavdevops2024 <bhargavdevops2024@gmail.com>
Date:   Tue Nov 26 21:08:04 2024 +0530

    Create m1.txt

commit 3d38733f9311c3045034d8600c47ee53797c15fd
Author: bhargavdevops2024 <bhargavdevops2024@gmail.com>
Date:   Tue Nov 26 21:03:53 2024 +0530

    Initial commit

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/demo-day2/lesson-end-project-lab2 (dev)
$ git status
On branch dev
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   index.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        c4
        c5
        c6


HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/demo-day2/lesson-end-project-lab2 (dev)
$ git add .

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/demo-day2/lesson-end-project-lab2 (dev)
$ git commit -m " c4 c5 c6 have been stagged and commited"
[dev 0b75ec5]  c4 c5 c6 have been stagged and commited
 4 files changed, 1 insertion(+)
 create mode 100644 c4
 create mode 100644 c5
```

```
HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/demo-day2/lesson-end-project-lab2 (dev)
$ git status
On branch dev
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   index.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        c4
        c5
        c6


HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/demo-day2/lesson-end-project-lab2 (dev)
$ git add .

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/demo-day2/lesson-end-project-lab2 (dev)
$ git commit -m " c4 c5 c6 have been stagged and commited"
[dev 0b75ec5]  c4 c5 c6 have been stagged and commited
 4 files changed, 1 insertion(+)
 create mode 100644 c4
 create mode 100644 c5
 create mode 100644 c6
 create mode 100644 index.txt

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/demo-day2/lesson-end-project-lab2 (dev)
$ git log
commit 0b75ec518c55e1ceea740bb2a9a9864ba5aaa7a5 (HEAD -> dev)
Author: bhargavdevops2024 <bhargavdevops2024@gmail.com>
Date:   Tue Nov 26 21:33:40 2024 +0530

     c4 c5 c6 have been stagged and commited

commit 09626f08eb8f0e37bc574f6bac5fe4fb866410aa (origin/main, origin/HEAD, main)
Author: bhargavdevops2024 <bhargavdevops2024@gmail.com>
Date:   Tue Nov 26 21:08:04 2024 +0530

    Create m1.txt

commit 3d38733f9311c3045034d8600c47ee53797c15fd
Author: bhargavdevops2024 <bhargavdevops2024@gmail.com>
Date:   Tue Nov 26 21:03:53 2024 +0530

    Initial commit

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/demo-day2/lesson-end-project-lab2 (dev)
```

Create c7 , git add . , commit and check git log 4 commits

```
HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/demo-day2/lesson-end-project-lab2 (dev)
$ touch c7

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/demo-day2/lesson-end-project-lab2 (dev)
$ git log
commit 0b75ec518c55e1ceea740bb2a9a9864ba5aaa7a5 (HEAD -> dev)
Author: bhargavdevops2024 <bhargavdevops2024@gmail.com>
Date:   Tue Nov 26 21:33:40 2024 +0530

     c4 c5 c6 have been stagged and commited

commit 09626f08eb8f0e37bc574f6bac5fe4fb866410aa (origin/main, origin/HEAD, main)
Author: bhargavdevops2024 <bhargavdevops2024@gmail.com>
Date:   Tue Nov 26 21:08:04 2024 +0530

    Create m1.txt

commit 3d38733f9311c3045034d8600c47ee53797c15fd
Author: bhargavdevops2024 <bhargavdevops2024@gmail.com>
Date:   Tue Nov 26 21:03:53 2024 +0530

    Initial commit

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/demo-day2/lesson-end-project-lab2 (dev)
$ git add .

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/demo-day2/lesson-end-project-lab2 (dev)
$ git commit -m "c7 added"
[dev 7008d57] c7 added
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 c7

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/demo-day2/lesson-end-project-lab2 (dev)
$ git log
commit 7008d574bf4e864a458c800007bba7efedddf3d6 (HEAD -> dev)
Author: bhargavdevops2024 <bhargavdevops2024@gmail.com>
Date:   Tue Nov 26 21:34:47 2024 +0530

    c7 added

commit 0b75ec518c55e1ceea740bb2a9a9864ba5aaa7a5
Author: bhargavdevops2024 <bhargavdevops2024@gmail.com>
Date:   Tue Nov 26 21:33:40 2024 +0530

     c4 c5 c6 have been stagged and commited
```

Check git log output and go back to master branch from dev and check git log

```
HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/demo-day2/lesson-end-project-lab2 (dev)
$ git log
commit 7008d574bf4e864a458c800007bba7efedddf3d6 (HEAD -> dev)
Author: bhargavdevops2024 <bhargavdevops2024@gmail.com>
Date:   Tue Nov 26 21:34:47 2024 +0530

    c7 added

commit 0b75ec518c55e1ceea740bb2a9a9864ba5aaa7a5
Author: bhargavdevops2024 <bhargavdevops2024@gmail.com>
Date:   Tue Nov 26 21:33:40 2024 +0530

     c4 c5 c6 have been stagged and commited

commit 09626f08eb8f0e37bc574f6bac5fe4fb866410aa (origin/main, origin/HEAD, main)
Author: bhargavdevops2024 <bhargavdevops2024@gmail.com>
Date:   Tue Nov 26 21:08:04 2024 +0530

    Create m1.txt

commit 3d38733f9311c3045034d8600c47ee53797c15fd
Author: bhargavdevops2024 <bhargavdevops2024@gmail.com>
Date:   Tue Nov 26 21:03:53 2024 +0530

    Initial commit

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/demo-day2/lesson-end-project-lab2 (dev)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/demo-day2/lesson-end-project-lab2 (main)
$ ls -ltr
total 2
-rw-r--r-- 1 HP 197121  5 Nov 26 21:10 m1.txt
-rw-r--r-- 1 HP 197121 25 Nov 26 21:10 README.md

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/demo-day2/lesson-end-project-lab2 (main)
$ git log
commit 09626f08eb8f0e37bc574f6bac5fe4fb866410aa (HEAD -> main, origin/main, origin/HEAD)
Author: bhargavdevops2024 <bhargavdevops2024@gmail.com>
Date:   Tue Nov 26 21:08:04 2024 +0530

    Create m1.txt

commit 3d38733f9311c3045034d8600c47ee53797c15fd
Author: bhargavdevops2024 <bhargavdevops2024@gmail.com>
```

Merge Dev Branch with Master .

- **git merge dev**

```
HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/demo-day2/lesson-end-project-lab2 (main)
$ git merge dev
Updating 09626f0..7008d57
Fast-forward
 c4        | 0
 c5        | 0
 c6        | 0
 c7        | 0
 index.txt | 1 +
 5 files changed, 1 insertion(+)
 create mode 100644 c4
 create mode 100644 c5
 create mode 100644 c6
 create mode 100644 c7
 create mode 100644 index.txt

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/demo-day2/lesson-end-project-lab2 (main)
$ git log
commit 7008d574bf4e864a458c800007bba7efedddf3d6 (HEAD -> main, dev)
Author: bhargavdevops2024 <bhargavdevops2024@gmail.com>
Date:   Tue Nov 26 21:34:47 2024 +0530

    c7 added

commit 0b75ec518c55e1ceea740bb2a9a9864ba5aaa7a5
Author: bhargavdevops2024 <bhargavdevops2024@gmail.com>
Date:   Tue Nov 26 21:33:40 2024 +0530

     c4 c5 c6 have been stagged and commited

commit 09626f08eb8f0e37bc574f6bac5fe4fb866410aa (origin/main, origin/HEAD)
Author: bhargavdevops2024 <bhargavdevops2024@gmail.com>
Date:   Tue Nov 26 21:08:04 2024 +0530

    Create m1.txt

commit 3d38733f9311c3045034d8600c47ee53797c15fd
Author: bhargavdevops2024 <bhargavdevops2024@gmail.com>
Date:   Tue Nov 26 21:03:53 2024 +0530

    Initial commit
```

Git Push Origin: Pushes the local changes to all the branches

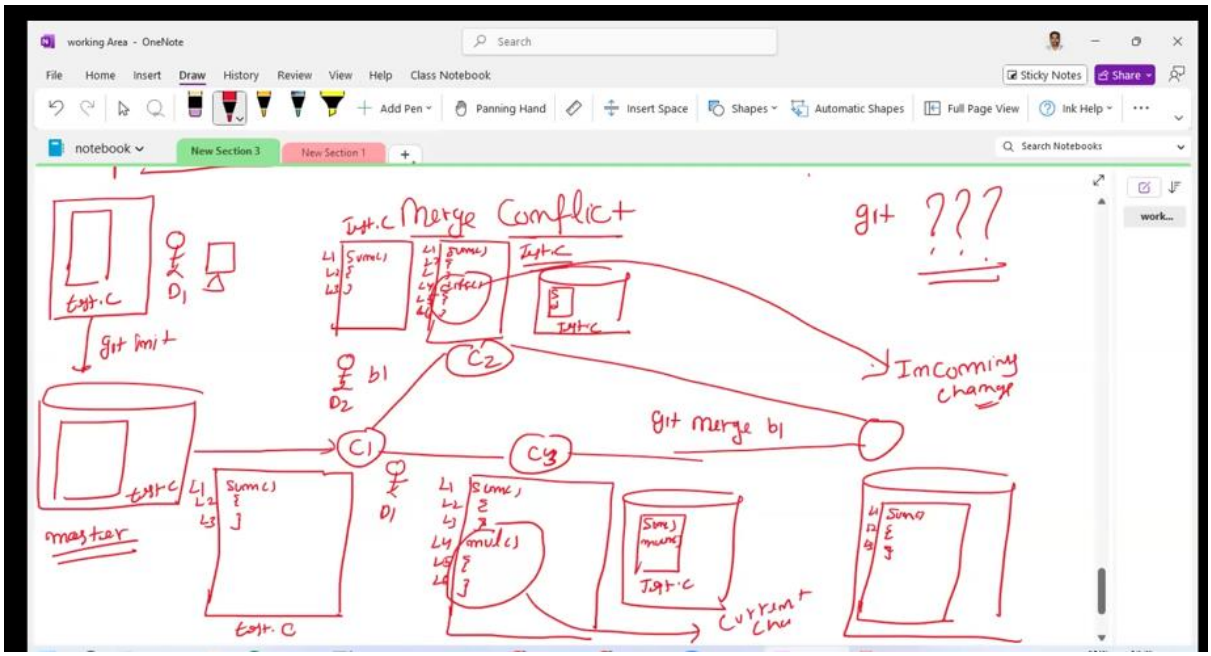
- **git push origin main : pushes changes to only main branch**
- **git push origin dev : pushes changes to only dev**

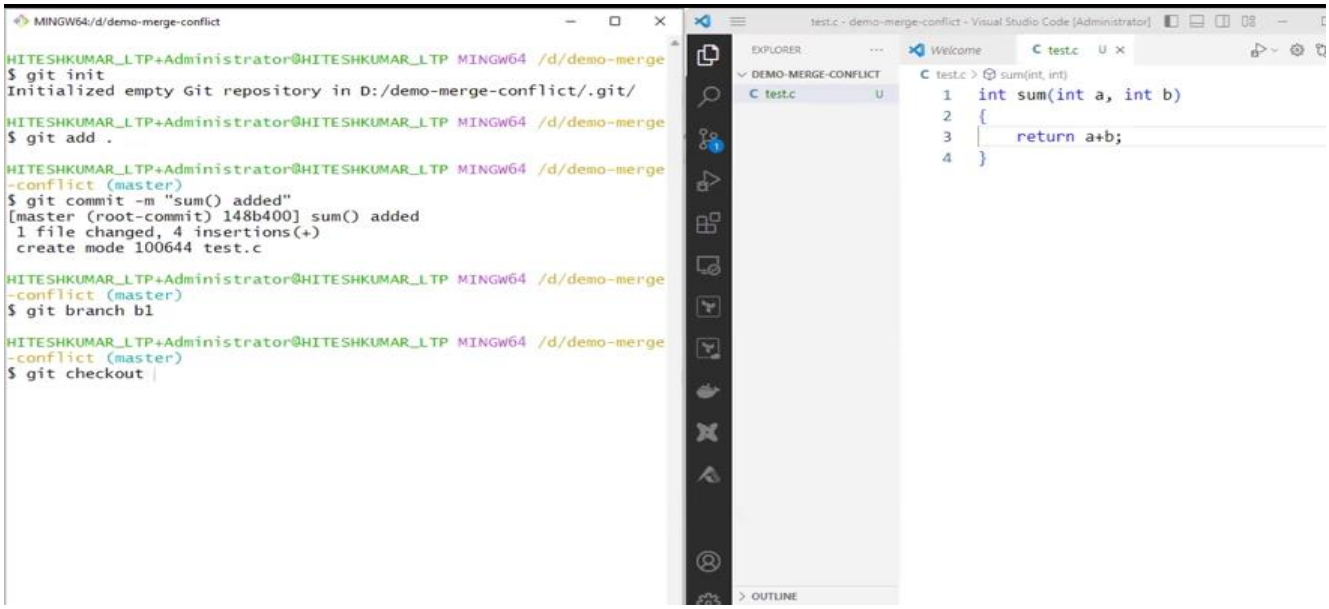
```
HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/demo-day2/lesson-end-project-lab2 (main)
$ git push origin
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 8 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (6/6), 537 bytes | 537.00 KiB/s, done.
Total 6 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/bhargavdevops2024/lesson-end-project-lab2.git
   09626f0..7008d57  main -> main

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/demo-day2/lesson-end-project-lab2 (main)
$ git push origin dev
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
remote: Create a pull request for 'dev' on GitHub by visiting:
remote:      https://github.com/bhargavdevops2024/lesson-end-project-lab2/pull/new/dev
remote:
To https://github.com/bhargavdevops2024/lesson-end-project-lab2.git
 * [new branch]      dev -> dev

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/demo-day2/lesson-end-project-lab2 (main)
$ git push origin main
Everything up-to-date
```

MERGE CONFLICT

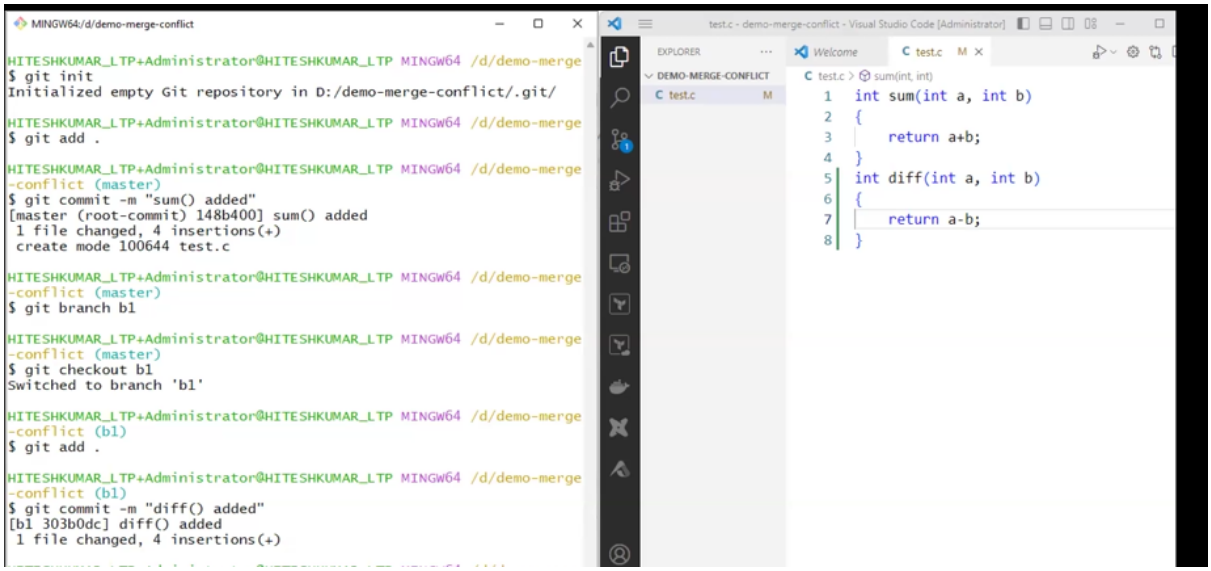
A merge conflict occurs when two or more developers make competing changes to the same file or line of code while working on a project:

- Two developers change the same lines in a file
- One developer deletes a file while another developer is modifying it
- A version of a file is newer than the version you started to base your changes on
- When a merge conflict occurs, Git will mark the file as conflicted and stop the merging process. The developers must then resolve the conflict.

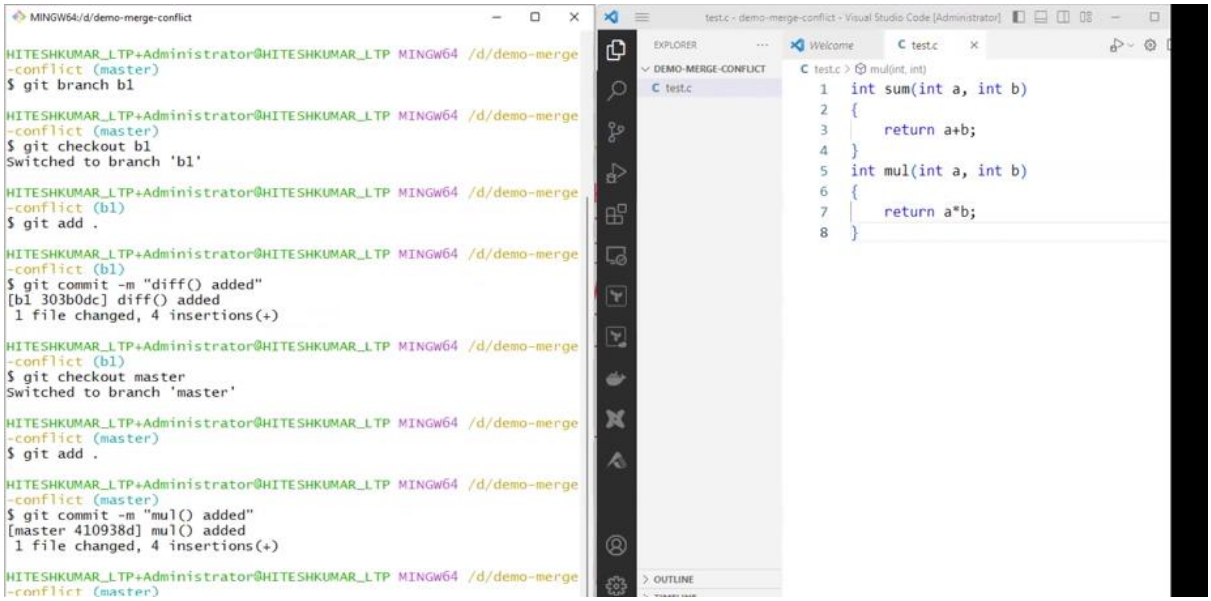




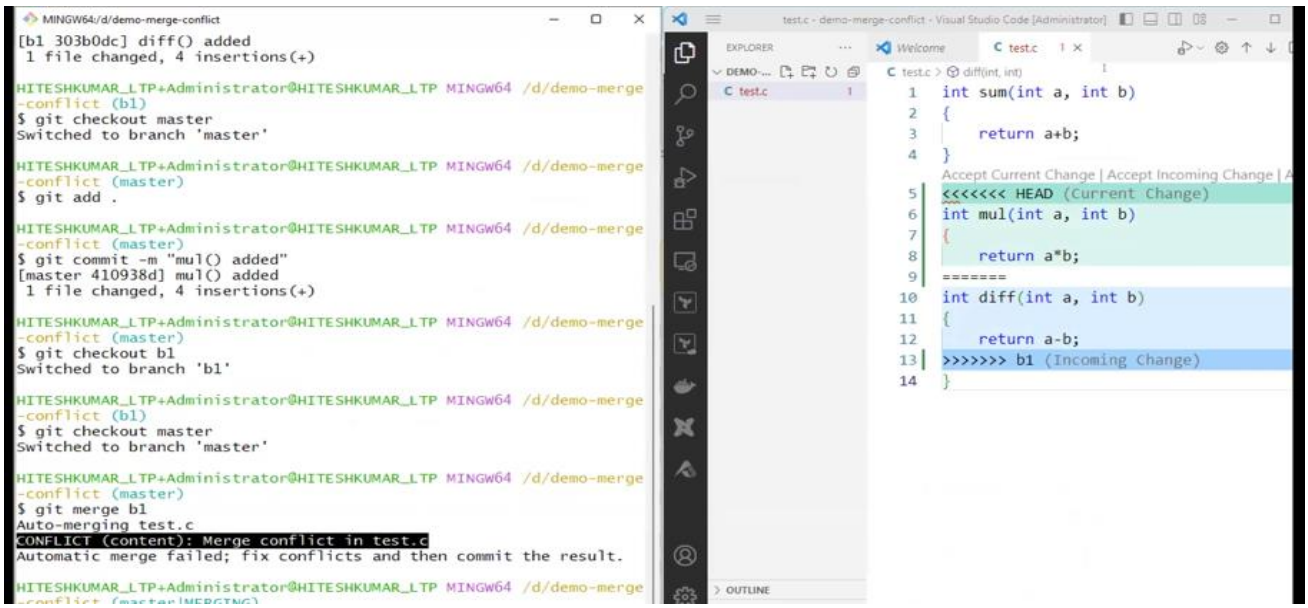
Checkout to Dev Branch and edit test.c file with diff function added.



Check Out to Master and add Multiplication in 3rd Commit

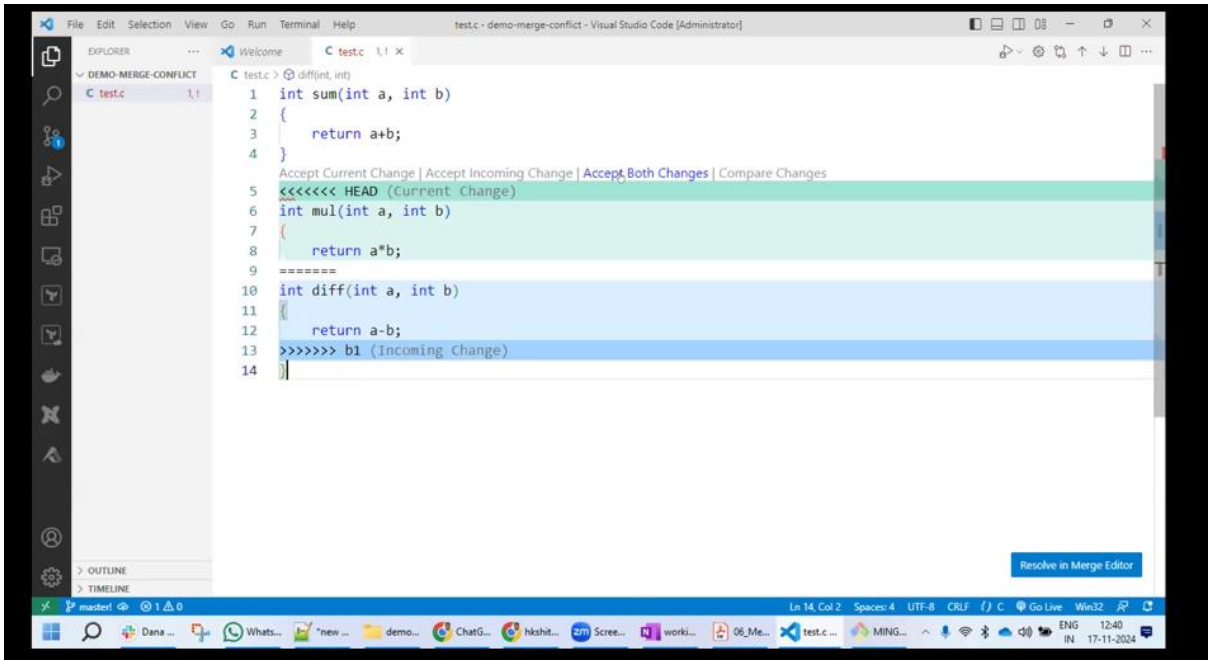


Merge Conflict Thrown:



Solution : In this case we need to take decision whether we need to

a. Accept Incoming Changes
b. Accept Current Change
c. Keep Both changes (Incoming changes added after other)



Lab : Merge Conflict

Step1: Create a file testc.txt in Master Branch

```
HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/demo-day2/lesson-end-project-lab2 (main)
$ cat testc.txt

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/demo-day2/lesson-end-project-lab2 (main)
$ cat testc.txt
hello ( int a , int b)

sum {

    return a+b

}
```

Step 2: Checkout to branch dev and add diff function as shown below to same testc.txt file

```
HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/demo-day2/lesson-end-project-lab2 (dev)
$ git commit -m "diff () added in dev branch"
On branch dev
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:   testc.txt

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

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/demo-day2/lesson-end-project-lab2 (dev)
$ cat testc.txt
hello ( int a , int b)

sum {

    return a+b

}

hello1 ( int c , int d)

diff {

   return a-b

}
```

Step 3: Now Switch Back to Main Branch and edit testc.txgt with mul() function

```
HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/demo-day2/lesson-end-project-lab2 (main)
$ cat testc.txt
hello ( int a , int b)

sum {

    return a+b

}

hello2 ( int c , int d)

mul {

   mul a*b

}
```

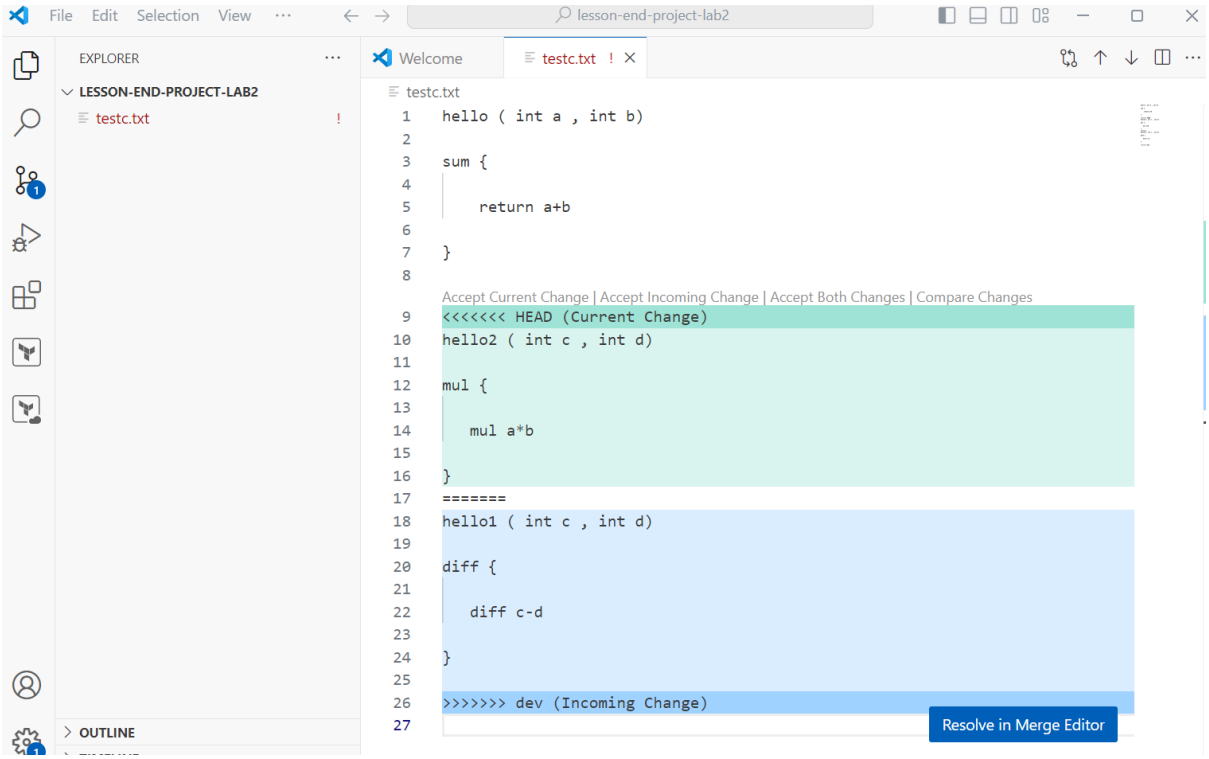
Step 4 : Now add and Commit. Lets Merge with Dev Branch changes and Merge conflict occurs

```
HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/demo-day2/lesson-end-project-lab2 (dev)
$ git add .
warning: in the working copy of 'testc.txt', LF will be replaced by CRLF the next time Git touches it

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/demo-day2/lesson-end-project-lab2 (dev)
$ git commit -m "diff() added"
[dev dc746bb] diff() added
 1 file changed, 16 insertions(+)

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/demo-day2/lesson-end-project-lab2 (dev)
$ git checkout main
Switched to branch 'main'
Your branch is ahead of 'origin/main' by 2 commits.
  (use "git push" to publish your local commits)

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/demo-day2/lesson-end-project-lab2 (main)
$ git merge dev
Auto-merging testc.txt
CONFLICT (content): Merge conflict in testc.txt
Automatic merge failed; fix conflicts and then commit the result.

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/demo-day2/lesson-end-project-lab2 (main|MERGING)
```

Step 5: Resolution

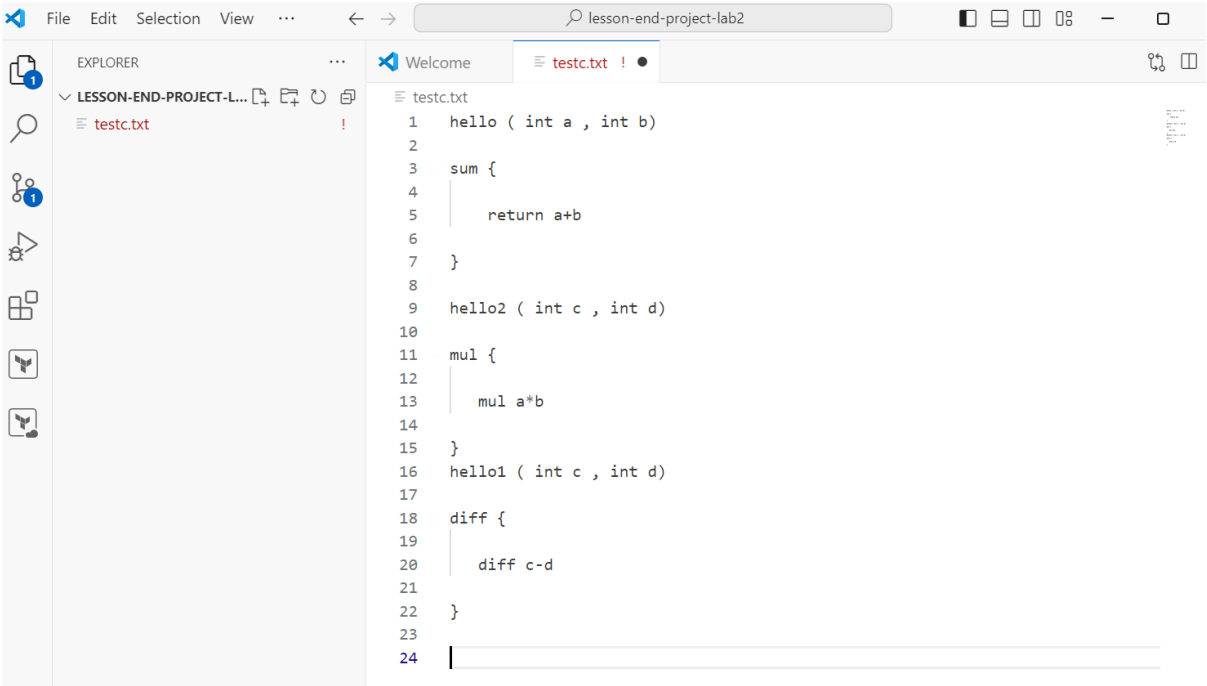
1. **Accept Incoming changes (From Dev Branch)**
2. **Accept Current Changes (From Main Branch)**

3. **Accept Both Changes (Dev & Main)**



Accepted Both Changes:

Output :

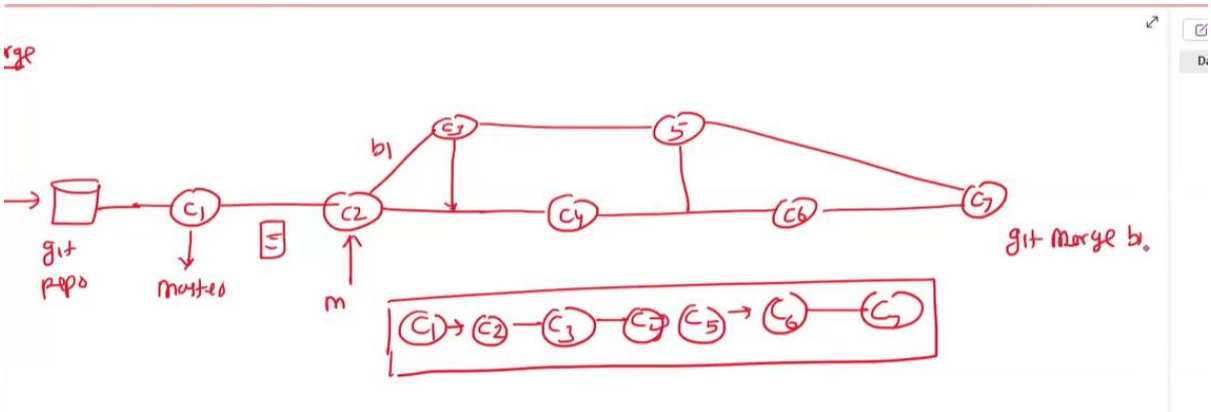


Incoming changes are added at last. If Only Incoming changes are accepted sum, diff () will be seen in testc.txt . if Current changes are accepted then sum,mul() are seen.

Note: Git wont be able to take any decision incase of Merge Conflict , either developer or Owner should take call regarding which changes are to be accepted.

GIT REBASE, GIT REVERT , GIT RESET

Git Merge Flow:



Sequence after Merge: c1-c2-c3-c4-c5-c6-c7

GIT REBASE:

Rebasing is changing the base of your branch from one commit to another making it appear as if you'd created your branch from a different commit. Internally, Git accomplishes this by creating new commits and applying them to the specified base.

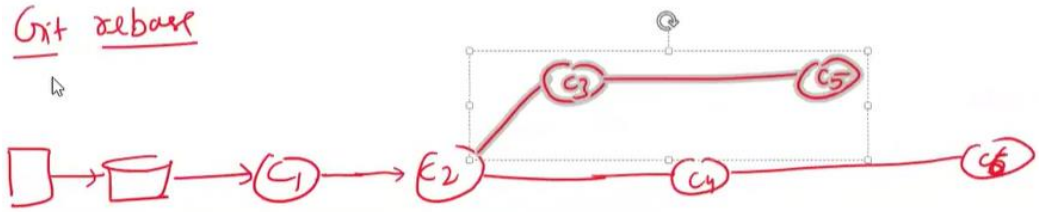
With the **rebase command**, you can take all the changes that were committed on one branch and replay them on a different branch.

The only **benefit** of rebase strategy is linear history and that's it.

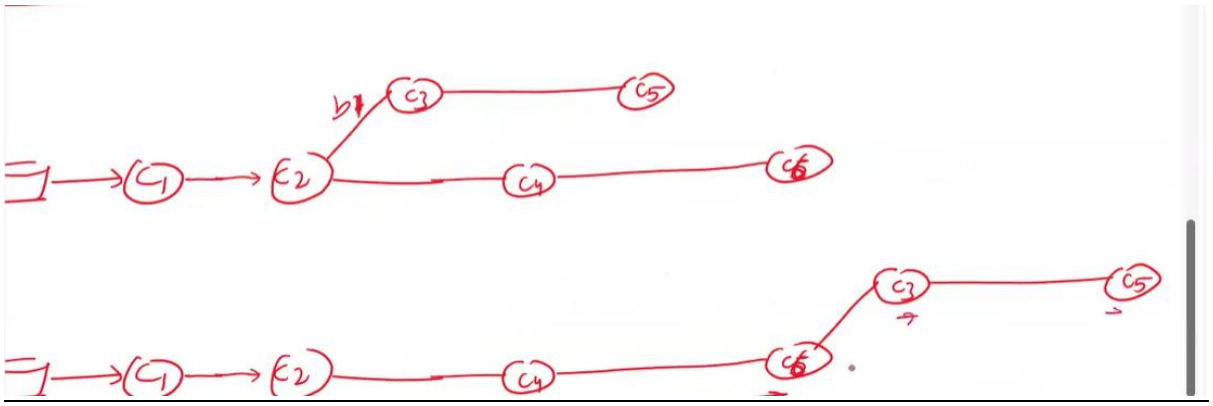
What is the difference between git merge and rebase?

Git Merge: Comparison. The main difference between git rebase and git merge is that git rebase creates a new set of commits applied on top of the target branch, while git merge creates a new merge commit that combines the changes from both branches. Allows users to merge branches in Git.

Normal Flow:



Rebased Flow:



Sequence : c1 – c2 – c4 – c6 – c3 – c5 : base has been changed

Advantage:

If we don't want changes that are made in the child branch (b1) . we can easily roll back to sequence c6 commit . Hence we can discard c3 and c5 changes done in new child branch.

Lab Exercise 3- Working with Git Rebase

Lab Exercise: Git Rebase

This exercise demonstrates the use of git rebase in a scenario where no conflicts occur.

Objective

1. Learn how to rebase branches when there are no conflicting changes.
2. Understand the clean, linear history created by git rebase.

Prerequisites

1. Install Git on your system.
2. Initialize a Git repository:

```
git init git-rebase-lab

cd git-rebase-lab
```

Step-by-Step Instructions

1. Set Up the Repository

1. Create the main branch and make the initial commit:

```
echo "Line 1 from main branch" > file.txt

git add file.txt
```

```
git commit -m "Initial commit: Add Line 1 from main branch"
```

2. Create a new branch feature-branch:

```
git checkout -b feature-branch
```

3. Add a new line to file.txt in feature-branch:

```
echo "Line 2 from feature branch" >> file.txt

git add file.txt

git commit -m "Add Line 2 from feature branch"
```

4. Switch back to the main branch and add another line:

```
git checkout main

echo "Line 3 from main branch" >> file.txt

git add file.txt

git commit -m "Add Line 3 from main branch"
```

---

2. Rebase feature-branch onto main

1. Switch to feature-branch:

```
git checkout feature-branch
```

2. Rebase feature-branch onto main:

```
git rebase main
```

3. Git will replay the commit from feature-branch onto the main branch. Since there are no conflicts, the rebase completes automatically.

---

3. Verify the Rebase

1. View the commit history:

```
git log --oneline --graph
```

Example output:

```
* Add Line 2 from feature branch

* Add Line 3 from main branch

* Initial commit: Add Line 1 from main branch
```

2. Check the contents of file.txt:

```
cat file.txt
```

Output:

```
Line 1 from main branch
```

Line 3 from main branch

Line 2 from feature branch

HandsOn:

On Master : Create file1.txt , file2.txt and commit with c1,c2 commits

```
HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/day3/rebase (main)
$ git init
Initialized empty Git repository in C:/Users/HP/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/day3/rebase/.git/

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/day3/rebase (master)
$ echo "Line 1 from main branch" > file.txt

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/day3/rebase (master)
$ git add .
warning: in the working copy of 'file.txt', LF will be replaced by CRLF the next time Git touches it

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/day3/rebase (master)
$ git status
On branch master

No commits yet

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


HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/day3/rebase (master)
$ git commit -m "c1 commit added file1"
[master (root-commit) 4569097] c1 commit added file1
 1 file changed, 1 insertion(+)
 create mode 100644 file.txt

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/day3/rebase (master)
$ echo "Line 2 from main branch" > file2.txt

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/day3/rebase (master)
$ git add .
warning: in the working copy of 'file2.txt', LF will be replaced by CRLF the next time Git touches it

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/day3/rebase (master)
$ git commit -m " commit c2 and added file2"
[master dd7f39e]  commit c2 and added file2
 1 file changed, 1 insertion(+)
 create mode 100644 file2.txt
```

Check git log –oneline output for two commits:

```
HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/day3/rebase (master)
$ git log --oneline
dd7f39e (HEAD -> master)  commit c2 and added file2
4569097 c1 commit added file1
```

Create and checkout to feature branch dev and create 1 more files test1.txt and commit it . we have c3 commit on dev branch

Check git log –online status for commits

```
HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/day3/rebase (master)
$ git checkout -b dev
Switched to a new branch 'dev'

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/day3/rebase (dev)
$ echo "Line 1 from dev branch" > test1.txt

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/day3/rebase (dev)
$ git add .
warning: in the working copy of 'test1.txt', LF will be replaced by CRLF the next time Git touches it

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/day3/rebase (dev)
$ git commit -m " test1.txt file from dev branch c3 commit"
[dev f139591]  test1.txt file from dev branch c3 commit
 1 file changed, 1 insertion(+)
 create mode 100644 test1.txt

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/day3/rebase (dev)
$ git log --oneline
f139591 (HEAD -> dev)  test1.txt file from dev branch c3 commit
dd7f39e (master)  commit c2 and added file2
4569097 c1 commit added file1
```

Now go back to master and create files file4.txt , file5.txt and commit it. C4 and c5 commits on master branch

```
HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/day3/rebase (master)
$ git log --oneline
dd7f39e (HEAD -> master)  commit c2 and added file2
4569097 c1 commit added file1

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/day3/rebase (master)
$ echo "Line 1 from main branch" > file3.txt

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/day3/rebase (master)
$ git add .
warning: in the working copy of 'file3.txt', LF will be replaced by CRLF the next time Git touches it

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/day3/rebase (master)
$ git commit -m "file3 added to main branch and it is c4 commit "
[master 3cbc494] file3 added to main branch and it is c4 commit
 1 file changed, 1 insertion(+)
 create mode 100644 file3.txt

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/day3/rebase (master)
$ echo "Line 1 from main branch" > file4.txt

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/day3/rebase (master)
$ git add .
warning: in the working copy of 'file4.txt', LF will be replaced by CRLF the next time Git touches it

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/day3/rebase (master)
$ git commit -m "file4 added to main branch and it is c5 commit "
[master 5194f80] file4 added to main branch and it is c5 commit
 1 file changed, 1 insertion(+)
 create mode 100644 file4.txt

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/day3/rebase (master)
$ git log --oneline
5194f80 (HEAD -> master) file4 added to main branch and it is c5 commit
3cbc494 file3 added to main branch and it is c4 commit
dd7f39e  commit c2 and added file2
4569097 c1 commit added file1
```

Checkout to dev branch and create test2.txt and test3.txt for c6 and c7 commits

```
HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/day3/rebase (master)
$ git checkout dev
Switched to branch 'dev'

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/day3/rebase (dev)
$ git log --oneline
f139591 (HEAD -> dev)  test1.txt file from dev branch c3 commit
dd7f39e  commit c2 and added file2
4569097 c1 commit added file1

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/day3/rebase (dev)
$ echo "Line 1 from main branch" > test2.txt

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/day3/rebase (dev)
$ echo "Line 1 from dev branch" > test2.txt

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/day3/rebase (dev)
$ echo "Line 3 from dev branch" > test3.txt

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/day3/rebase (dev)
$ git add .
warning: in the working copy of 'test2.txt', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'test3.txt', LF will be replaced by CRLF the next time Git touches it

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/day3/rebase (dev)
$ git commit -m " test2.txt and 3.txt have been added for c6 and c7 commits"
[dev e0e9b1c]  test2.txt and 3.txt have been added for c6 and c7 commits
 2 files changed, 2 insertions(+)
 create mode 100644 test2.txt
 create mode 100644 test3.txt

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/day3/rebase (dev)
$ git log --oneline
e0e9b1c (HEAD -> dev)  test2.txt and 3.txt have been added for c6 and c7 commits
f139591  test1.txt file from dev branch c3 commit
dd7f39e  commit c2 and added file2
4569097 c1 commit added file1
```

Apply Rebase to merge dev branch sequence first with Master commits

$ git rebase dev master

Successfully rebased and updated refs/heads/master.

```
HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/day3/rebase (dev)
$ git rebase dev master
Successfully rebased and updated refs/heads/master.

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/day3/rebase (master)
$ git log
commit c49e617cecacdb6e97088b71987759ddb4fe149a (HEAD -> master)
Author: bhargavdevops2024 <bhargavdevops2024@gmail.com>
Date:   Wed Nov 27 12:56:12 2024 +0530

    file4 added to main branch and it is c5 commit

commit 5d0efab0c6a35e1b6bfe37bd1b90602b690f7d1c
Author: bhargavdevops2024 <bhargavdevops2024@gmail.com>
Date:   Wed Nov 27 12:55:26 2024 +0530

    file3 added to main branch and it is c4 commit

commit e0e9b1cc094d9688ab967e8758c7f93077e20af9 (dev)
Author: bhargavdevops2024 <bhargavdevops2024@gmail.com>
Date:   Wed Nov 27 13:00:19 2024 +0530

     test2.txt and 3.txt have been added for c6 and c7 commits

commit f13959147c016e82d81e08cc28afea68d5f1936a
Author: bhargavdevops2024 <bhargavdevops2024@gmail.com>
Date:   Wed Nov 27 12:52:58 2024 +0530

     test1.txt file from dev branch c3 commit

commit dd7f39e1e5ac25f95bdae4262b39ff5b53bc33bf
Author: bhargavdevops2024 <bhargavdevops2024@gmail.com>
Date:   Wed Nov 27 12:47:03 2024 +0530

     commit c2 and added file2

commit 45690977d739a114f4648c09bc73eeb2f611251f
Author: bhargavdevops2024 <bhargavdevops2024@gmail.com>
Date:   Wed Nov 27 12:46:01 2024 +0530
```

Sequence : c1 -> c2 -> c3 -> c6->c7->c4->c5 } Merged

C1,C2: Initial Commits

C3,C6,C7 : commits from feature dev branch

C5,C5: commits from Base Main Branch

Sequence when you are in DEV Branch and Merge with Master: Initial Commits – Dev Branch Commits – Main Branch Commits

b) When you are In Main branch and rebase with Dev :

git rebase main dev

C1 ->C2 ->C4->C5->C3->C6->C7

C4,C5 : Main Branch Commits first

C3,C6,C7 : Dev Branch commits next

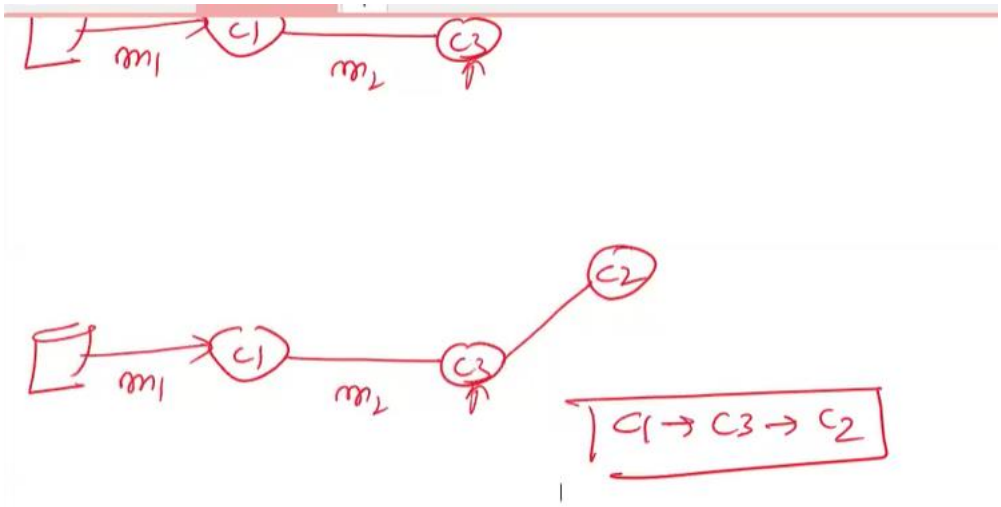
Merge: git merge dev master:

```
HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/day3/rebase (master)
$ git checkout dev
Switched to branch 'dev'

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/day3/rebase (dev)
$ git merge
fatal: No remote for the current branch.

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/day3/rebase (dev)
$ git merge master
Updating e0e9b1c..c49e617
Fast-forward
 file3.txt | 1 +
 file4.txt | 1 +
 2 files changed, 2 insertions(+)
 create mode 100644 file3.txt
 create mode 100644 file4.txt

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/day3/rebase (dev)
$ git log --oneline
c49e617 (HEAD -> dev, master) file4 added to main branch and it is c5 commit
5d0efab file3 added to main branch and it is c4 commit
e0e9b1c  test2.txt and 3.txt have been added for c6 and c7 commits
f139591  test1.txt file from dev branch c3 commit
dd7f39e  commit c2 and added file2
4569097 c1 commit added file1

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/day3/rebase (dev)
```

C1,c2,c3,c6,c7,c4,c5

Git Rebase Continue : Inorder to Skip incase of any Merge conflicts

Another Simple Example



HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/day3 (main)

$ mkdir rebase2

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/day3 (main)

$ cd rebase2

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/day3/rebase2 (main)

$ git init

Initialized empty Git repository in C:/Users/HP/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/day3/rebase2/.git/

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/day3/rebase2 (master)

$ touch m1

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/day3/rebase2 (master)

$ git add .

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/day3/rebase2 (master)

$ git commit -m "c1 commit"

[master (root-commit) a240afe] c1 commit

1 file changed, 0 insertions(+), 0 deletions(-)

create mode 100644 m1

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/day3/rebase2 (master)

$ git checkout -b dev

Switched to a new branch 'dev'

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/day3/rebase2 (dev)

$ touch b1

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/day3/rebase2 (dev)

$ git add .

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/day3/rebase2 (dev)

$ git commit -m "c2 commit"

[dev 9e3c0a6] c2 commit

1 file changed, 0 insertions(+), 0 deletions(-)

create mode 100644 b1

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/day3/rebase2 (dev)

$ git checkout master

Switched to branch 'master'

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/day3/rebase2 (master)

$ touch m2

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/day3/rebase2 (master)

$ git add .

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/day3/rebase2 (master)

$ git commit -m "c3 commit"

[master 67c664d] c3 commit

1 file changed, 0 insertions(+), 0 deletions(-)

create mode 100644 m2

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/day3/rebase2 (master)

$ git checkout dev

Switched to branch 'dev'

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/day3/rebase2 (dev)

$ git rebase master

Successfully rebased and updated refs/heads/dev.

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/day3/rebase2 (dev)

$ git log --online

fatal: unrecognized argument: --online

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/day3/rebase2 (dev)

$ git log --oneline

8914f89 (HEAD -> dev) c2 commit

67c664d (master) c3 commit

a240afe c1 commit

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/day3/rebase2 (dev)

$ git merge master

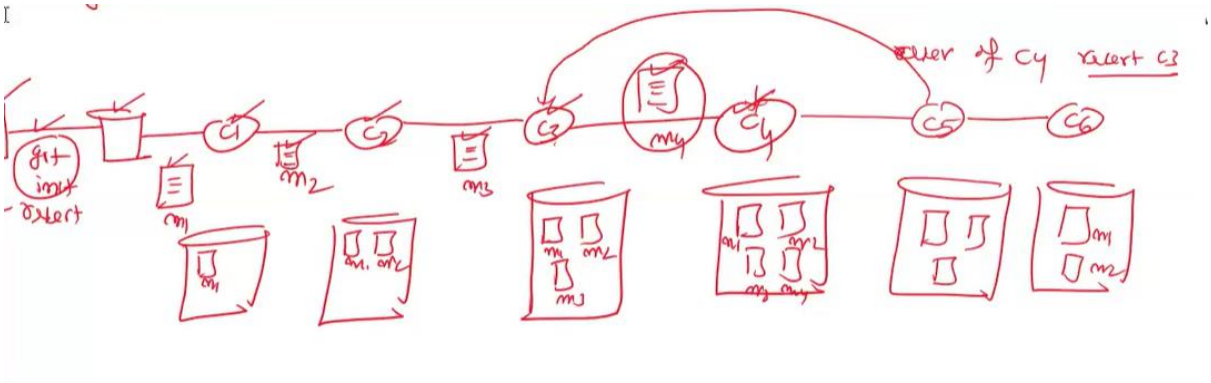
Already up to date.

➢ GIT REVERT

The git revert command is used for undoing changes to a repository's commit history. Other 'undo' commands like, git checkout and git reset, move the HEAD and branch ref pointers to a specified commit. Git revert also takes a specified commit, however, git revert does not move ref pointers to this commit

Example: If there are 10 commits are if you want changes of 8th commit and need to go back to the contents of that commit we will use git revert.

- **Git log gives you all commit ID''s, you can revert back to the commit you need'**
- **git revert 1239516d27818889cfc40e2a5ec9bd6ec634aeba**



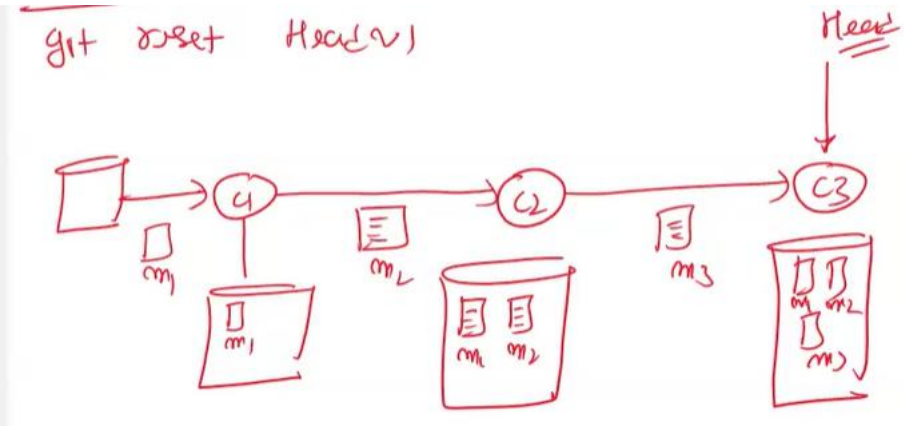

```
HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/demo-day2/lesson-end-project-lab2 (main)
$ git log
commit f859ae695ab3ab4812e2ad47f93c6561f0714aec (HEAD -> main)
Author: bhargavdevops2024 <bhargavdevops2024@gmail.com>
Date:   Wed Nov 27 00:30:46 2024 +0530

    Revert "1st commit"

    This reverts commit 596ca20acdb7892366f2c4af50dc7158e66e6d86.

commit ddc4178182a6429254f545a47112bb054a44319a
Author: bhargavdevops2024 <bhargavdevops2024@gmail.com>
Date:   Wed Nov 27 00:30:10 2024 +0530

    2nd commit

commit 596ca20acdb7892366f2c4af50dc7158e66e6d86
Author: bhargavdevops2024 <bhargavdevops2024@gmail.com>
Date:   Wed Nov 27 00:28:50 2024 +0530

    1st commit

commit 1239516d27818889cfc40e2a5ec9bd6ec634aeba
Author: bhargavdevops2024 <bhargavdevops2024@gmail.com>
Date:   Wed Nov 27 00:27:50 2024 +0530

    1st commit

commit 3ff8ba60f69d4bb17a97473eaf3980fd252e65cb
Author: bhargavdevops2024 <bhargavdevops2024@gmail.com>
Date:   Wed Nov 27 00:25:31 2024 +0530

    Revert "testc file with sum() added"

    This reverts commit 1cf456b064eccc8105952f6f5cb04ae51c2681b1.

commit 164af2bf2fb1bd93829483a2164b2da2c47ce621
Merge: 41d5a2d dc746bb
Author: bhargavdevops2024 <bhargavdevops2024@gmail.com>
Date:   Wed Nov 27 00:24:45 2024 +0530

    new change added
```

```
HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/demo-day2/lesson-end-project-lab2 (main)
$ git revert 1239516d27818889cfc40e2a5ec9bd6ec634aeba
[main 4319bb0] Revert "1st commit"
 7 files changed, 3 insertions(+)
 create mode 100644 README.md
 rename hello.txt => c4 (100%)
 create mode 100644 c5
 create mode 100644 c6
 create mode 100644 c7
 create mode 100644 index.txt
 create mode 100644 m1.txt

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/demo-day2/lesson-end-project-lab2 (main)
$ ls
1  README.md  c4  c5  c6  c7  hello2.txt  index.txt  m1.txt

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/demo-day2/lesson-end-project-lab2 (main)
$ 5~
```

Now we got back the changes of the commit ID we have reverted.

GIT RESET

Git Reset. reset is the command we use when we want to move the repository back to a previous commit , discarding any changes made after that commit .



Diff between Git Revert and Git Reset:

Both are used to revert or rollback to previous changes but if we use git revert then a new snapshot of commit is created . Even after rollback to certain commit , we will have a new commit where we can again rollback to the actual commit .

Git Revert:

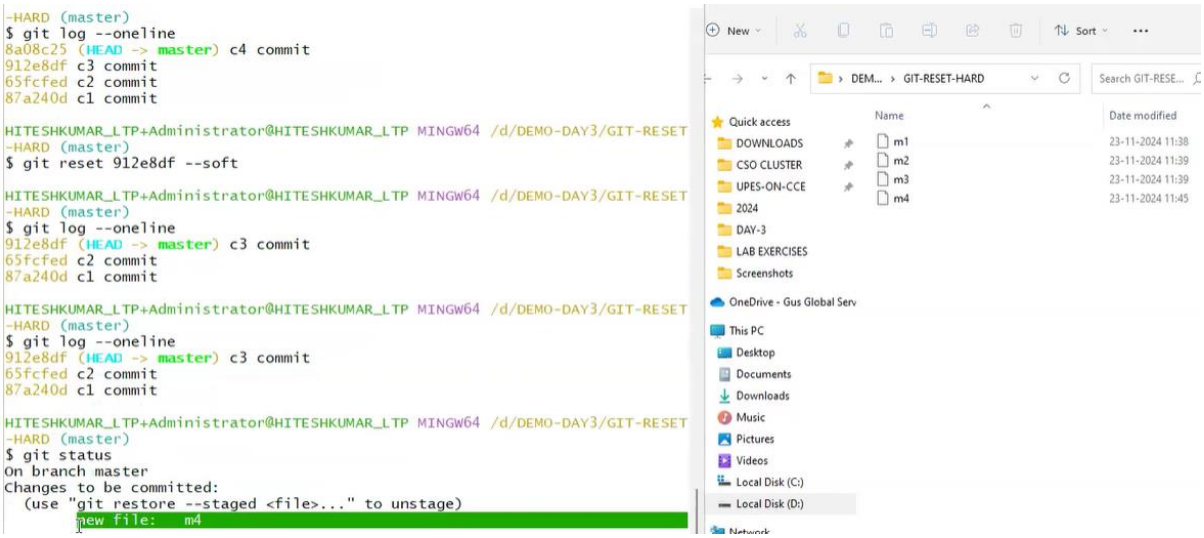
C1->C2->C3->C4 ---- If need to use revert to c3 we will rollback to c3 but a new commit c5 will be created with all the c4 changes. So when we need to get c4 changes again we can again revert to c5 commit. Therefore we will all changes in c4 as usual

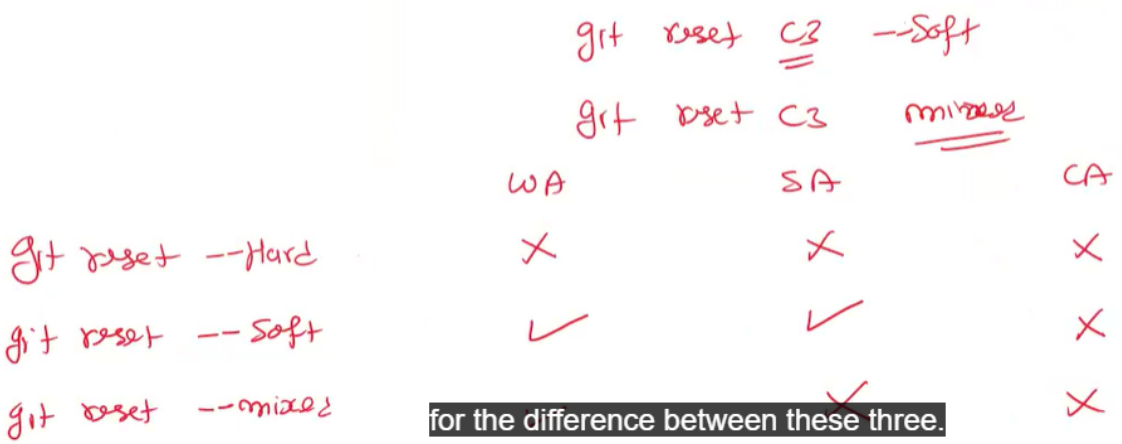
Git Reset: When we reset to C3 all the commits after c3 will be deleted and we don't have any commits further.

Hence we cannot rollback to c4 commit again as we don't have a snapshot.

Soft Reset:

C1->C2->C3->C4 : git reset c3 : Head points to the c3 commit but we have file stored in Staged area. We can again commit it to bring back to c4 commit.





Mixed: Only in working area,need to stage and commit again to get c4 commit.

Reset Mixed: File present in Work Area, Need to be staged and commited again

```
HITESHKUMAR_LTP+Administrator@HITESHKUMAR_LTP MINGW64 /d/DEMO-DAY3/GIT-RESET
-HARD (master)
$ git log --oneline
5d301eb (HEAD -> master) c4 commit
912e8df c3 commit
65fcfed c2 commit
87a240d c1 commit

HITESHKUMAR_LTP+Administrator@HITESHKUMAR_LTP MINGW64 /d/DEMO-DAY3/GIT-RESET
-HARD (master)
$ git reset 912e8df --mixed

HITESHKUMAR_LTP+Administrator@HITESHKUMAR_LTP MINGW64 /d/DEMO-DAY3/GIT-RESET
-HARD (master)
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        m4

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

HITESHKUMAR_LTP+Administrator@HITESHKUMAR_LTP MINGW64 /d/DEMO-DAY3/GIT-RESET
-HARD (master)
$ git log --oneline
912e8df (HEAD -> master) c3 commit
65fcfed c2 commit
87a240d c1 commit
```

Handson:

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/day3 (main)

$ mkdir reset

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/day3 (main)

$ cd reset

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/day3/reset (main)

$ git init

Initialized empty Git repository in C:/Users/HP/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/day3/reset/.git/

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/day3/reset (master)

$ touch f1

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/day3/reset (master)

$ git add .

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/day3/reset (master)

$ git commit -m "c1 commit"

[master (root-commit) 974c604] c1 commit

1 file changed, 0 insertions(+), 0 deletions(-)

create mode 100644 f1

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/day3/reset (master)

$ touch f2

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/day3/reset (master)

$ git add .

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/day3/reset (master)

$ git commit -m "c2 commit"

[master 58b533f] c2 commit

1 file changed, 0 insertions(+), 0 deletions(-)

create mode 100644 f2

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/day3/reset (master)

$ touch c32

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/day3/reset (master)

$ git add .

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/day3/reset (master)

$ git commit -m "c3"

[master dc99070] c3

1 file changed, 0 insertions(+), 0 deletions(-)

create mode 100644 c32

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/day3/reset (master)

$ git log

commit dc990706e667099f26f400cfa9feef565cc5f10b (HEAD -> master)

Author: bhargavdevops2024 <bhargavdevops2024@gmail.com>

Date: Wed Nov 27 18:09:23 2024 +0530

c3

commit 58b533f58ba78a4ed80b0a7ea0e97c2b3703409a

Author: bhargavdevops2024 <bhargavdevops2024@gmail.com>

Date: Wed Nov 27 18:09:00 2024 +0530

c2 commit

commit 974c604afbca97951193175683b7a361b899334b

Author: bhargavdevops2024 <bhargavdevops2024@gmail.com>

Date: Wed Nov 27 18:08:31 2024 +0530

c1 commit

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/day3/reset (master)

$ git log --oneline

dc99070 (HEAD -> master) c3

58b533f c2 commit

974c604 c1 commit

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/day3/reset (master)

$ git reset 58b533f58ba78a4ed80b0a7ea0e97c2b3703409a --soft

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/day3/reset (master)

$ git status

On branch master

Changes to be committed:

(use "git restore --staged <file>..." to unstage)

new file: c32

The most similar command is

commit

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/day3/reset (master)

$ git commit -m "c3 commit and file again brought to commit stage"

[master 570d429] c3 commit and file again brought to commit stage

1 file changed, 0 insertions(+), 0 deletions(-)

create mode 100644 c32

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/day3/reset (master)

$ git log --oneline

570d429 (HEAD -> master) c3 commit and file again brought to commit stage

58b533f c2 commit

974c604 c1 commit

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/day3/reset (master)

$ git reset 58b533f58ba78a4ed80b0a7ea0e97c2b3703409a --mixed

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/day3/reset (master)

$ git status

On branch master

Untracked files:

(use "git add <file>..." to include in what will be committed)

c32

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

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/day3/reset (master)

$ git add .

git c

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/day3/reset (master)

$ git commit -m "C3 commit , file brought again to staged and commited again"

[master 5b4183a] C3 commit , file brought again to staged and commited again

1 file changed, 0 insertions(+), 0 deletions(-)

create mode 100644 c32

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/day3/reset (master)

$ git log --oneline

5b4183a (HEAD -> master) C3 commit , file brought again to staged and commited again

58b533f c2 commit

974c604 c1 commit

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/day3/reset (master)

$ git reset 58b533f58ba78a4ed80b0a7ea0e97c2b3703409a --hard

HEAD is now at 58b533f c2 commit

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/day3/reset (master)

$ git log --oneline

58b533f (HEAD -> master) c2 commit

974c604 c1 commit

PULL REQUEST (PR)

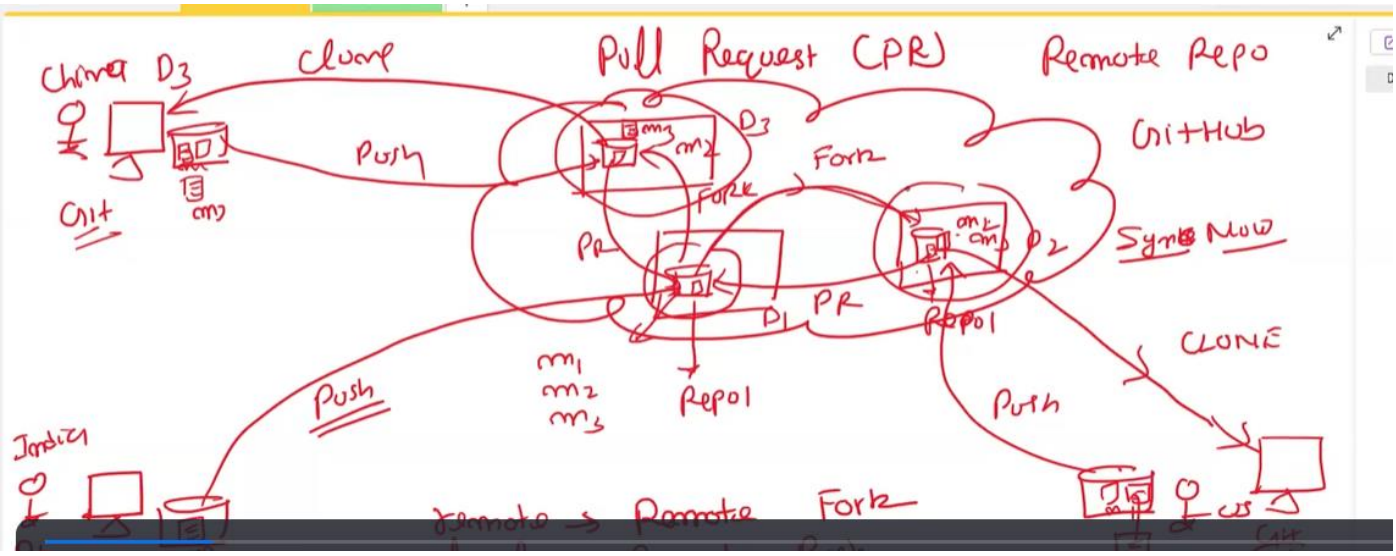
A pull request, often abbreviated as PR, serves as a proposal to merge changes made in one branch of a repository into another, typically from a feature branch into the main branch. Pull requests are essential for facilitating code reviews, encouraging collaboration, and maintaining a clean, well-documented codebase.

Remote -> Remote : Fork

Local -> Remote : Push

Remote -> Remote : PR

Initially the Repository in Remote account account is forked and changes are made. Once the changes are done the other user/developer will make a Pull Request to the Owner to accept the changes.Once the PR is accepted the changes are merged and the person who pushes the changes becomes the Contributor for the project



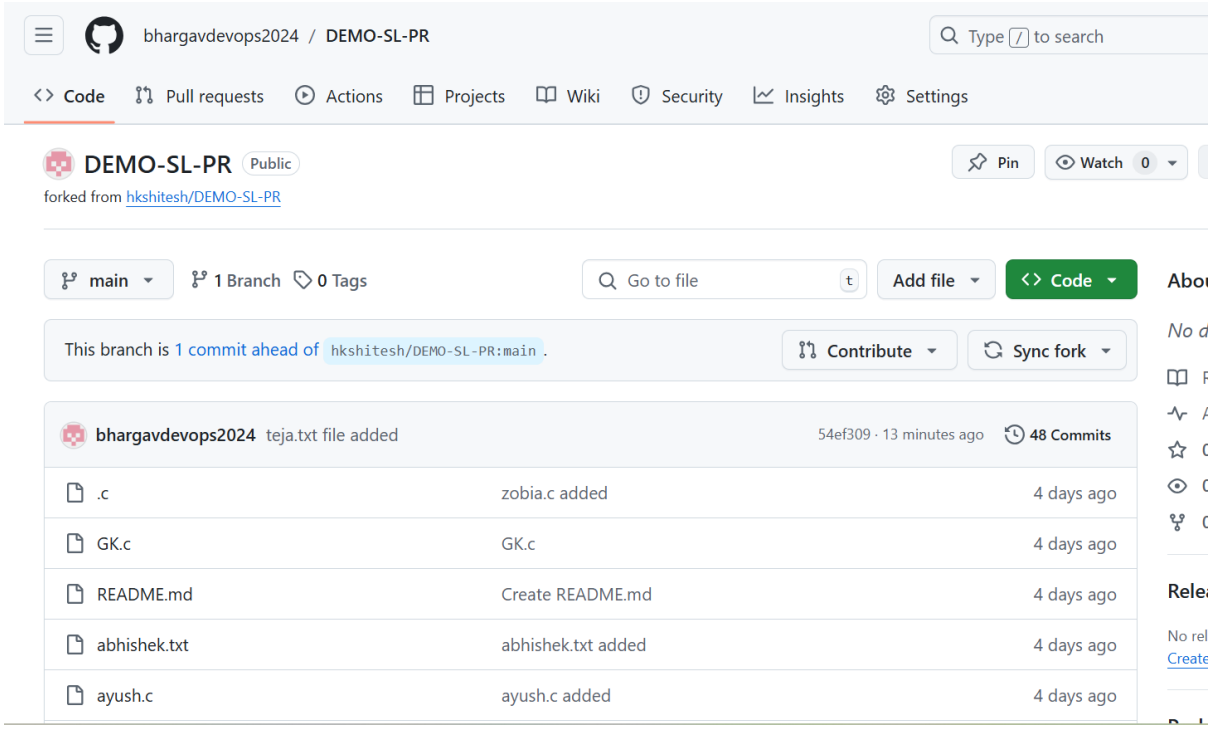
Clone the Remote Repo of hitesh/Demo-SL-PR.git

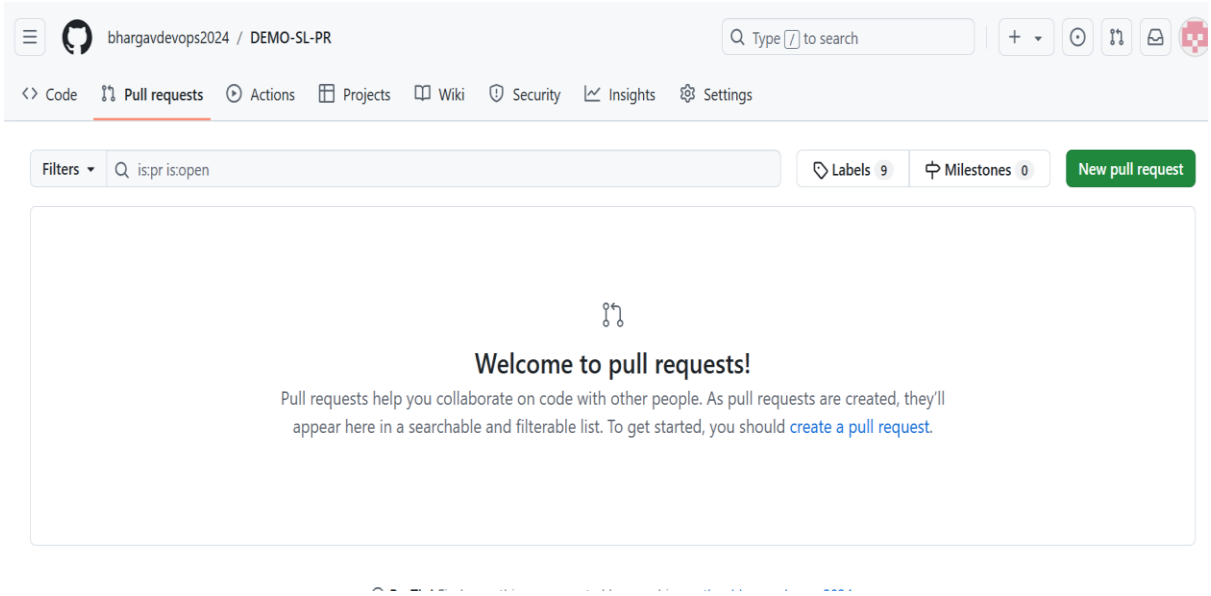
```
HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/day3/Fork-Demo (main)
$ git clone https://github.com/bhargavdevops2024/DEMO-SL-PR.git
Cloning into 'DEMO-SL-PR'...
remote: Enumerating objects: 93, done.
remote: Counting objects: 100% (53/53), done.
remote: Compressing objects: 100% (39/39), done.
remote: Total 93 (delta 28), reused 20 (delta 13), pack-reused 40 (from 1)
Receiving objects: 100% (93/93), 25.78 KiB | 8.59 MiB/s, done.
Resolving deltas: 100% (28/28), done.
```

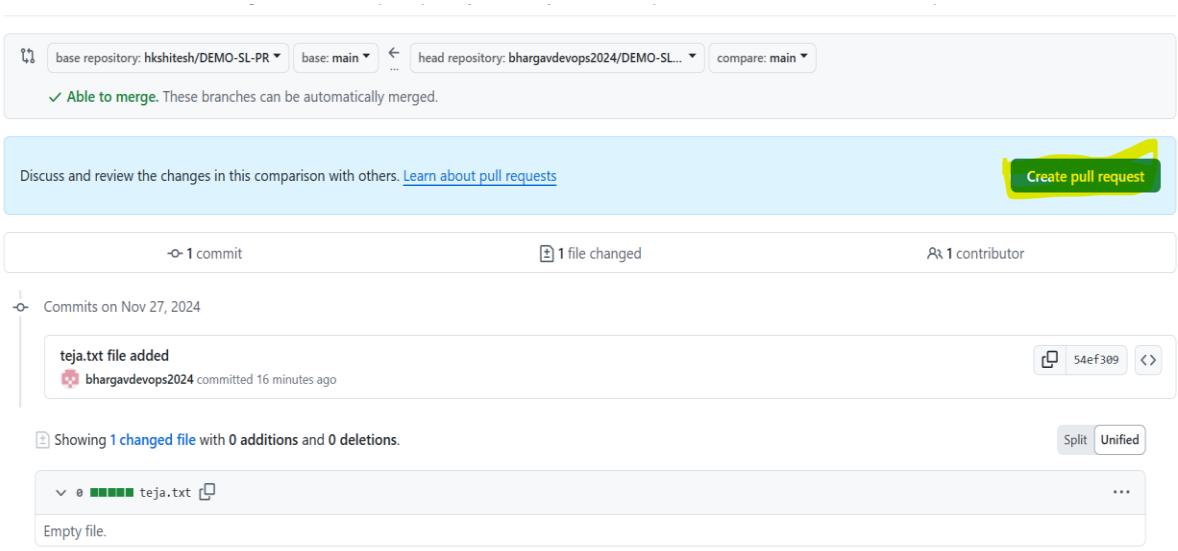
Create a new file on local and push it to the Demo-SLR repo with in your account

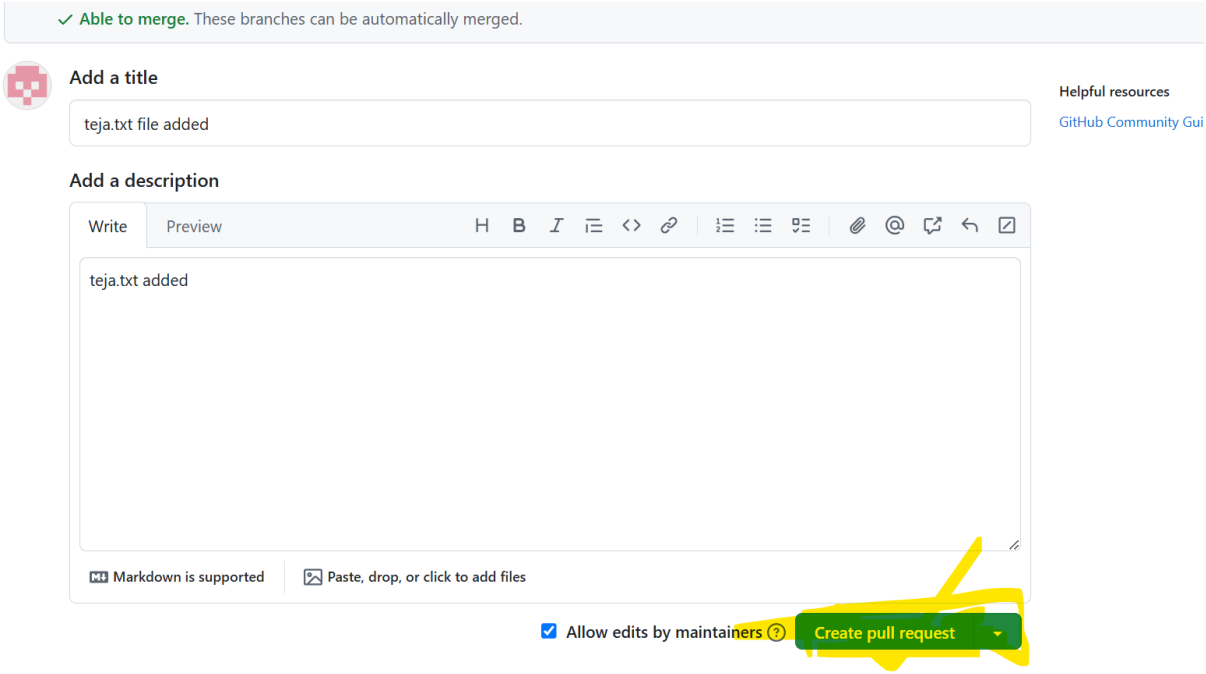
```
HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/day3/Fork-Demo (main)
$ cd DEMO-SL-PR/

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/day3/Fork-Demo/DEMO-SL-PR (main)
$ ls
GK.c  README.md  abhishek.txt  ayush.c  dk.c  hitesh.c  hks.c  m1  maruthi  poorna.c  samidh.c  shandilya.c  sid.c  testreddy  vaishu.c  wajid.c  zobia  zobia.c

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/day3/Fork-Demo/DEMO-SL-PR (main)
$ touch teja.txt

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/day3/Fork-Demo/DEMO-SL-PR (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

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

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

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/day3/Fork-Demo/DEMO-SL-PR (main)
$ git add .

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/day3/Fork-Demo/DEMO-SL-PR (main)
$ git commit  -m "teja.txt file added"
[main 54ef309] teja.txt file added
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 teja.txt

HP@DESKTOP-AMSJ8IS MINGW64 ~/Desktop/Devops/GitHubLabs/Day1/CI-CD-Project/day3/Fork-Demo/DEMO-SL-PR (main)
$ git push
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 240 bytes | 240.00 KiB/s, done.
Total 2 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/bhargavdevops2024/DEMO-SL-PR.git
   fac1f40..54ef309  main -> main
```

Generate a Pull Request



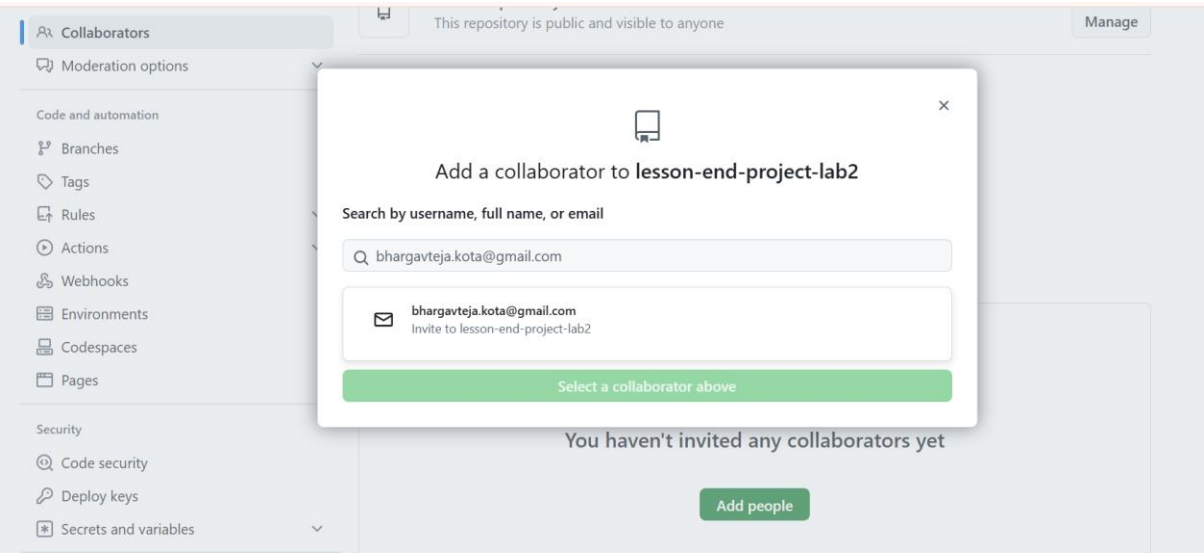






Once the Owner receives the pull request he reviews the changes and accepts the PR . Once the PR is accepted the code merges with the repository of Owner account. Now the developer who pushes the code becomes the contributor of Project from now.

Colloborator: When we want to provide owner priviligers to others,we can make them as collaborator. Now the user who gets that access can accepts PR and Merge the PR’s



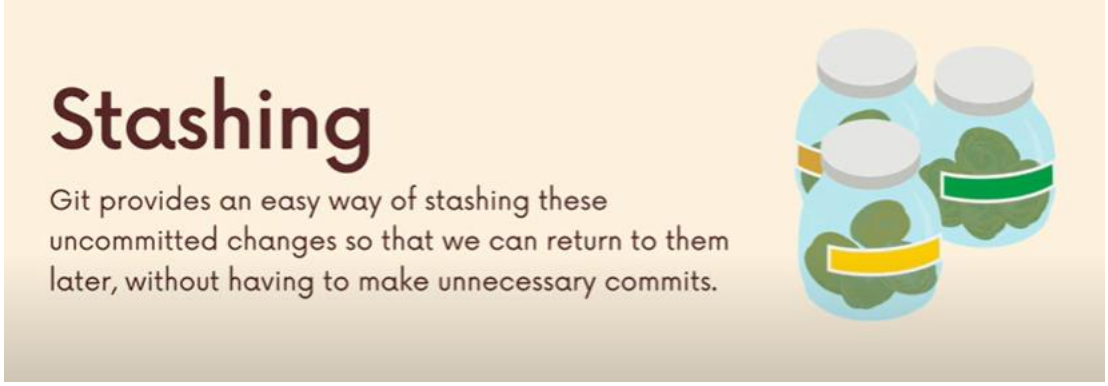
Once the email invitation is accepts the user becomes the co-owner of the Repository.

MERGE CONFLICT IN GIT PR

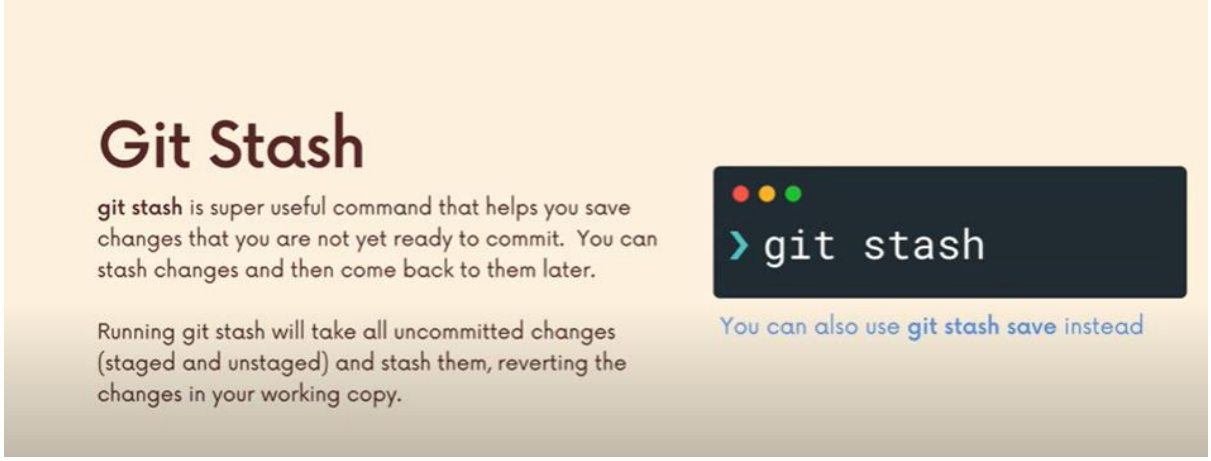
Conflicts generally arise when two people have changed the same lines in a file, or if one developer deleted a file while another developer was modifying it. In these cases, Git cannot automatically determine what is correct.

- First Multiple users in different Git Accounts fork the Repository of the Owner
- Once the Repository is cloned , they may make some changes to the existing file in both accounts.
- Once the change are made they push the changes to the forked Repo in their account from Local.
- Once the changes are available in Local , they make a PR (Pull Request) to Owner Account.
- Sometimes when both make changes to same file with some differences in both the file make pull request then a conflict occurs when the owner is trying to merge
- Git cannot handle the conflict and Owner takes the call, whether to accept contributor 1 changes or contributor2 changes or both.
- Next the PR is Merged.

GIT STASH



SWITCHING FROM FEATURE BRANCH WHERE YOU HAVE UNCOMMITED WORK TO MAIN :



ONCE YOUR WORK ON MAIN BRANCH COMPLETES SWITCH BACK TO FEATURE BRANCH AND USE GIT STASH POP TO COMEOUT OF STASH STATE :

Stashing

Use **git stash pop** to remove the most recently stashed changes in your stash and re-apply them to your working copy.

```
❯ git stash pop
```

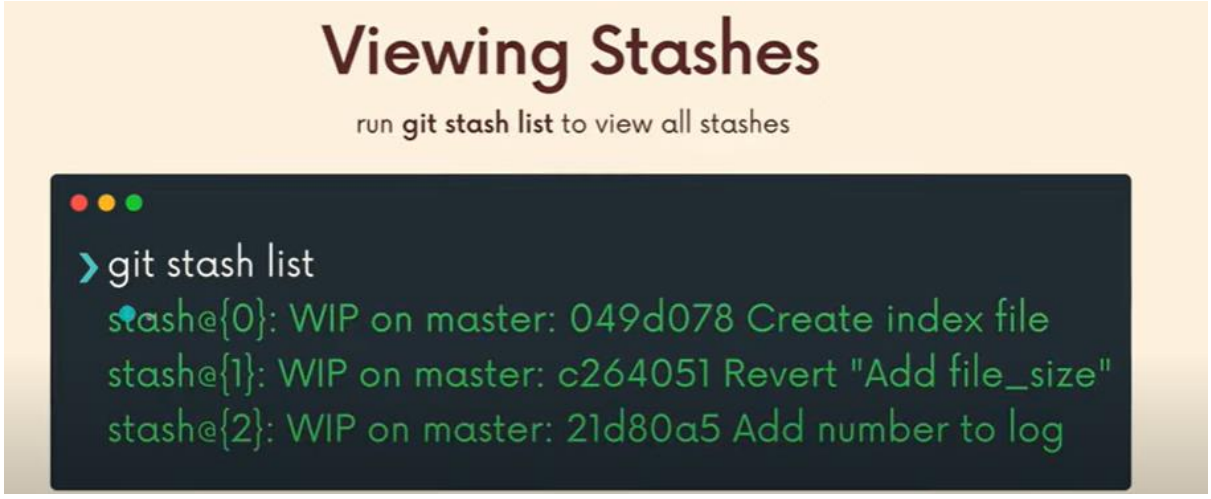
```
en by checkout:
        main.py
Please commit your changes or stash them before you switch branches
.
Aborting
(base) ImportantProject ❯ git stash                          feat1
Saved working directory and index state WIP on feat1: ce65d4e add f
eature1
(base) ImportantProject ❯ git switch main                    feat1
Switched to branch 'main'
(base) ImportantProject ❯ git switch feat1                    main
Switched to branch 'feat1'
(base) ImportantProject ❯ git stash pop                      feat1
On branch feat1
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working direct
ory)
        modified:   main.py

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

NOW USE GIT STASH APPLY TO APLLY THE CHANGES:

Stash Apply

You can use **git stash apply** to apply whatever is stashed away, without removing it from the stash. This can be useful if you want to apply stashed changes to multiple branches.

```
❯ git stash apply
```



THEORY:

Git Stash temporarily shelves (or *stashes*) changes you've made to your working copy so you can work on something else, and then come back and re-apply them later on.

Example: You are working on Feature Branch made some files , staged or unstaged them then you are asked to go to master and quickly verify something. You wont be able to switch from the Feature branch to Main Branch with out commiting changes . In this case we use Stash Apply to temporarily save the changes ,switch back to master do your work and come back.

Stashing your work

The git stash command takes your uncommitted changes (both staged and unstaged), saves them away for later use, and then reverts them from your working copy. For example:

```
$ git status
On branch main
Changes to be committed:

    new file:   style.css

Changes not staged for commit:

    modified:   index.html

$ git stash
Saved working directory and index state WIP
HEAD is now at 5002d47 our new homepage

$ git status
On branch main
nothing to commit, working tree clean
```

At this point you're free to make changes, create new commits, switch branches, and perform any other Git operations; then come back and re-apply your stash when you're ready.

Note that the stash is local to your Git repository; stashes are not transferred to the server when you push.

Re-applying your stashed changes

You can reapply previously stashed changes with git stash pop

```
$ git status
On branch main
nothing to commit, working tree clean
$ git stash pop
On branch main
Changes to be committed:

    new file:   style.css

Changes not staged for commit:

    modified:   index.html

Dropped refs/stash@{0} (32b3aa1d185dfe6d57b3c3cc3b32cbf3e380cc6a)
```

Popping your stash removes the changes from your stash and reapplies them to your working copy.

Alternatively, you can reapply the changes to your working copy *and* keep them in your stash with git stash apply:

with `git stash apply`:

```
$ git stash apply
On branch main
Changes to be committed:

    new file:   style.css

Changes not staged for commit:

    modified:   index.html
```

This is useful if you want to apply the same stashed changes to multiple branches.

Now that you know the basics of stashing, there is one caveat with git stash you need to be aware of: by default Git *won't* stash changes made to untracked or ignored files.