**Objectives**

**Explain branching and merging**

Branching and Merging in Git

* Branching: In Git, a branch is a separate line of development that allows you to work on new features, bug fixes, or experiments without affecting the main codebase. The default branch is often called master or main.  
  Example: git checkout -b feature-branch
* Merging: Merging combines changes from one branch into another (often from a feature branch into master). Git tries to automatically integrate changes, but if the same parts of files were changed in both branches, merge conflicts must be resolved manually.  
  Example: git merge feature-branch

**Explain about creating a branch request in GitLab**

Creating a Branch Request in GitLab

* In GitLab, a “branch request” isn’t a formal Git term, but it usually means **creating a new branch in the remote repository** so other team members can see and work on it.
* You can create a branch:
  1. Locally using git checkout -b branch-name and pushing it:  
     git push -u origin branch-name
  2. Or directly in GitLab’s web interface by selecting “New Branch” from the repository page.

**Explain about creating a merge request in GitLab**

Creating a Merge Request in GitLab

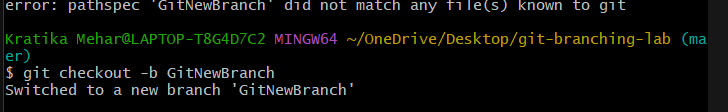
* A Merge Request (MR) in GitLab is a request to merge changes from one branch into another (e.g., from feature-branch into master).
* Steps (short version):
  1. Push your branch to GitLab.
  2. Go to Merge Requests in the GitLab project.
  3. Click New merge request.
  4. Select source branch (your feature branch) and target branch (e.g., master).
  5. Add title/description and submit.
* Team members can review, discuss, and approve before the merge happens.

**Steps**

**Part 1 – Branching**

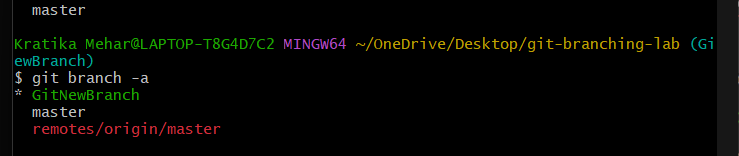
**1. Create a new branch named GitNewBranch**

git checkout -b GitNewBranch

****

**2. List all local and remote branches**

git branch -a

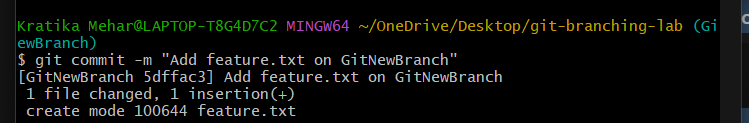


**3. Add a new file and commit changes**

echo "This is a new feature" > feature.txt

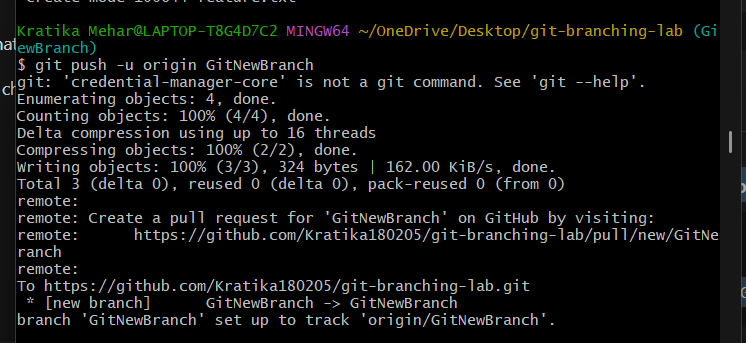
git add feature.txt

git commit -m "Add feature.txt on GitNewBranch"



**4. Push branch to remote**

git push -u origin GitNewBranch



**Part 2 – Creating a Merge Conflict for P4Merge**

**1. Edit the same file differently on both branches**

**On master:**

git checkout master

echo "This is master version" > feature.txt

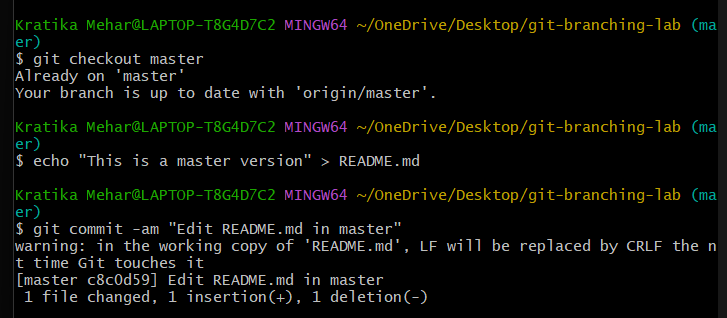
git commit -am "Edit feature.txt in master"

**On GitNewBranch:**

git checkout GitNewBranch

echo "This is branch version" > feature.txt

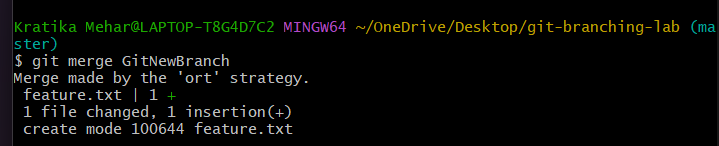
git commit -am "Edit feature.txt in branch"



**2. Merge to cause conflict**

git checkout master

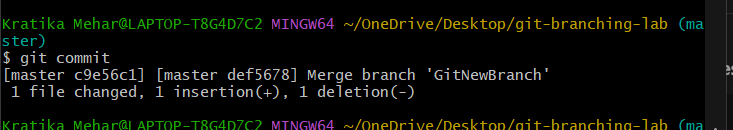
git merge GitNewBranch



**3. Stage and commit resolved file**

git add feature.txt

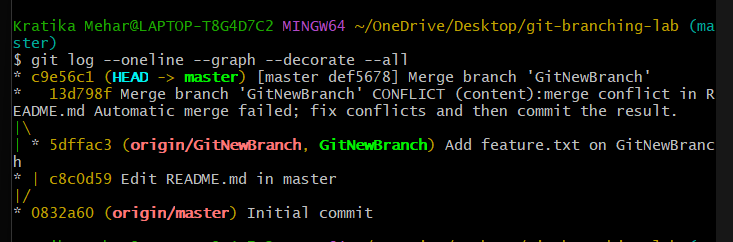
git commit



**Part 3 – Final Steps**

**1. View merge history**

git log --oneline --graph --decorate –all



**2. Delete branch locally and remotely**

git branch -d GitNewBranch

git push origin --delete GitNewBranch

