Git Branching & Merging – Full Hands-on Step-by-Step

1. Explain Branching and Merging

Branching in Git allows you to create a separate line of development from the main codebase (main or master branch).  
This is useful for:

* Developing new features
* Fixing bugs
* Experimenting with changes

Each branch is an isolated environment, so your work doesn’t affect the stable code.

Merging is the process of combining changes from one branch (like a feature branch) into another branch (usually main or master).

There are two types:

* Fast-forward merge: When the main branch hasn’t moved forward
* Three-way merge: When both branches have diverged

Merging brings your changes into the official codebase once they’re reviewed and tested.

2. Explain About Creating a Branch Request in GitLab?

Steps to Create a Branch in GitLab:

1. Log in to [GitLab.com](https://gitlab.com)
2. Go to your project repository
3. On the left menu, click Repository → Branches
4. Click the “New branch” button
5. Enter:
   * Branch name (e.g., feature-login)
   * Create from: usually main or master
6. Click “Create branch”

A new branch is now created and can be used for making changes separately from the main branch.

3)Explain About Creating a Merge Request in GitLab

A Merge Request (MR) in GitLab is how you ask someone to review and merge your branch into another branch (usually main or master).

Steps to Create a Merge Request:

1. Go to your project on GitLab
2. Navigate to Merge Requests from the left sidebar
3. Click “New Merge Request”
4. Choose:
   * Source branch (e.g., GitNewBranch)
   * Target branch (e.g., master)
5. Click “Compare branches and continue”
6. Fill in:
   * Title (e.g., “Add login feature”)
   * Description (what was changed)
7. Click “Create merge request”

Now:

* Reviewers can review the code
* You or they can merge the branch into the target after approval

Branching

Goal:

Create a new branch GitNewBranch, make some changes, and commit them.

Step-by-Step:

1. Open Git Bash in your project folder

Make sure you're in the folder where Git is initialized (you've already run git init earlier):

cd /your/project/path

2. Create a new branch called GitNewBranch

git branch GitNewBranch

This creates the new branch but doesn’t switch to it yet.

3. List all local branches

git branch

\* master

GitNewBranch

The \* indicates the current active branch (still master now).

4. Switch to the new branch

git checkout GitNewBranch

Now you are working inside the GitNewBranch branch.

5. Add a new file in this branch

Create a file:

echo "This is the branch file" > branchfile.txt

6. Stage the file

git add branchfile.txt

7. Commit the change

git commit -m "Added branchfile.txt in GitNewBranch"

8. Check status

git status

Output should show a clean working directory.

Part 2: Merging

Goal:

Merge changes from GitNewBranch into master.

Step-by-Step:

1. Switch back to the master branch

git checkout master

2. Check for differences (optional CLI view)

git diff GitNewBranch

This shows what’s different between master and GitNewBranch.

3. Visual diff with P4Merge (if configured)

If P4Merge is installed and configured as your Git diff tool:

git difftool GitNewBranch

It will open a visual difference viewer.

4. Merge GitNewBranch into master

git merge GitNewBranch

You should see a success message:

Updating abc123..def456

Fast-forward

branchfile.txt | 1 +

1 file changed, 1 insertion(+)

5. View merge history

git log --oneline --graph --decorate

You'll see a visual history of commits and merges.

6. Delete the branch after merging

git branch -d GitNewBranch

-d deletes the branch only if it's already merged into the current branch (master).

7. Check status again

git status

It should show a clean working directory, and your branchfile.txt should now exist in master.

Summary of Commands

# Branching

git branch GitNewBranch

git branch

git checkout GitNewBranch

echo "This is the branch file" > branchfile.txt

git add branchfile.txt

git commit -m "Added branchfile.txt in GitNewBranch"

git status

# Merging

git checkout master

git diff GitNewBranch

git difftool GitNewBranch # If P4Merge is configured

git merge GitNewBranch

git log --oneline --graph --decorate

git branch -d GitNewBranch

git status