**Explain branching and merging**

Branching is like making a **parallel copy** of your code so you can work on changes **without affecting the main code**.

**Purpose:**

* + Work on a new feature
  + Fix a bug
  + Experiment without breaking the main branch

Example :-

# Create a new branch

git branch feature-login

# Switch to it

git checkout feature-login

# Or both in one command

git checkout -b feature-login

**Merging in Git**

Merging means **combining changes from one branch into another**.

* **Example:**  
  You finish your feature in feature-login and want it in main:

# Switch to the branch you want to merge into

git checkout main

# Merge the other branch

git merge feature-login

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

In GitLab, you can create a branch in **two main ways**:

**A) From the Web UI**

1. Go to your GitLab repository.
2. Click on **Repository → Branches**.
3. Click **New Branch**.
4. Enter branch name (e.g., feature-login).
5. Choose source branch (usually main or develop).
6. Click **Create Branch**.

**B) From Local Git**

git checkout -b feature-login

git push -u origin feature-login

After pushing, you’ll see the branch in GitLab.

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

A Merge Request (MR) is how you propose merging changes into another branch (like main).  
It’s similar to a Pull Request in GitHub.

**Steps (from GitLab UI):**

1. Go to your project in GitLab.
2. If you’ve just pushed a branch, GitLab will show **“**Create merge request**”** → click it.  
   Or go to Merge Requests **→** New Merge Request.
3. Select:
   * Source branch (your feature branch)
   * Target branch (main or the branch you want to merge into)
4. Fill in:
   * **Title** (short summary of the changes)
   * Description (what changes were made, why)
5. Assign reviewers (if needed).
6. Click Create merge request.
7. Reviewers can comment, request changes, or approve.
8. Once approved → click Merge.

**Branching Steps**

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

git branch GitNewBranch

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

git branch -a

The \* shows your current active branch.

**3. Switch to the new branch:**

git checkout GitNewBranch

*(or newer syntax:)*

git switch GitNewBranch

**4. Add a new file and some content:**

echo "This is content for the new branch" > branch\_file.txt

**5. Stage the file:**

git add branch\_file.txt

**6. Commit the file to the branch:**

git commit -m "Add branch\_file.txt in GitNewBranch"

**7. Check branch status:**

git status

**Merging Steps**

**1. Switch to the main branch (instead of master):**

git checkout main

*(or)*

git switch main

**2. List differences between main and GitNewBranch:**

git diff main GitNewBranch

**3. Show visual differences with P4Merge (if configured):**

git mergetool

If P4Merge is not configured, run:

git config --global merge.tool p4merge

git config --global mergetool.path "C:/Program Files/Perforce/p4merge.exe"

Then re-run:

git mergetool

**4. Merge GitNewBranch into main:**

git merge GitNewBranch

**5. View commit history in a graph:**

git log --oneline --graph --decorate

**6. Delete the merged branch locally:**

git branch -d GitNewBranch

**7. (Optional) Delete branch from remote:**

git push origin --delete GitNewBranch

**8. Verify branches:**

git branch -a

**9. Final status check:**

git status







