Git-Branch

Last updated by | shakibesaib | Aug 31, 2023 at 4:53 PM GMT+6

Branching in **Git** allows us to create independent lines of development within a repository. It's useful for working on different features, bug fixes, or experiments without affecting the main codebase until we're ready to merge the changes.

Branching workflow example:

1. `git branch` : Lists all existing branches in the repository.
2. `git checkout -b feature-branch` : Creating new branch feature-branch and switching to it.
3. Make changes, add files, and commit them on the feature branch:
   ```
   git add <file1> <file2>
   git commit -m "Implement feature XYZ"
   ```
4. `git checkout main` : Switch back to the main branch.
5. `git merge feature-branch` : Merge the changes from the feature branch into the main branch
6. Delete the feature branch : `git branch -d feature-branch`

Common Branching Strategies:

1. **Feature Branches**:

- Create a branch for each new feature or task.
- Work on the feature and commit regularly.
- Merge the feature branch back into the main branch when it's ready.

2. **Release Branches**:

- Create a release branch from the main branch when preparing for a new release.
- Bug fixes can be applied to both the release and main branches.

3. **Hotfix Branches**:

- Create a hotfix branch to fix critical issues in the main branch.
- Merge the hotfix back into the main branch and any active release branches.