Table of Contents

[Introduction 2](#_Toc169726399)

[Step-by-Step Guide 2](#_Toc169726400)

[Creating and Switching Branches 2](#_Toc169726401)

[Making Changes and Committing 2](#_Toc169726402)

[Merging Branches 2](#_Toc169726403)

[Resolve Conflicts 3](#_Toc169726404)

[Deleting the Merged Branch 3](#_Toc169726405)

[Introduction 3](#_Toc169726406)

[Step-by-Step Guide 3](#_Toc169726407)

[Forking a Repository 3](#_Toc169726408)

[Clone the Forked Repository: 3](#_Toc169726409)

[Making Changes and Committing 3](#_Toc169726410)

[Stage and Commit Changes: 4](#_Toc169726411)

[Creating a Pull Request 4](#_Toc169726412)

[Conclusion 4](#_Toc169726413)

Branching and Merging in Git: A Local Guide

# Introduction

Git is a powerful version control system that enables developers to collaborate on projects efficiently. Two fundamental concepts in Git are branching and merging. Branching allows you to diverge from the main line of development and continue to work without affecting the main branch. Merging brings together changes from different branches. This report will guide you through creating branches, making changes, and merging branches locally using Git and GitHub.

# Step-by-Step Guide

## Creating and Switching Branches

* Create a New Branch:

git branch new-branch

* Switch to the New Branch:

git checkout new-branch

* Alternatively, you can create and switch to a new branch in one command:

git checkout -b new-branch

## Making Changes and Committing

* Make Changes: Modify files or add new files in your project directory.
* Stage Changes:

git add .

* Commit Changes

git commit -m "Your commit message"

## Merging Branches

* Switch Back to the Main Branch:

git checkout main

* Merge the New Branch into Main:

git merge new-branch

Resolve Conflicts: If there are any conflicts, Git will notify you. Open the affected files, resolve the conflicts, then stage and commit the resolved files:

git add .

git commit -m "Resolved merge conflicts"

## Deleting the Merged Branch

* Delete the Branch:

git branch -d new-branch

Forking and Pull Requests: A GitHub Guide

# Introduction

Forking a repository on GitHub allows you to create a copy of the repository under your GitHub account. This is useful for contributing to projects you do not have write access to. After making changes in your fork, you can submit a pull request to the original repository, suggesting your changes. This report will guide you through forking a repository, making changes, and submitting a pull request.

# Step-by-Step Guide

## Forking a Repository

Fork the Repository:

* Navigate to the repository on GitHub you want to fork.
* Click the "Fork" button in the top-right corner of the repository page.

## Clone the Forked Repository:

Copy the URL of your forked repository. Clone it to your local machine.

git clone https://github.com/yourusername/your-forked-repository.git

Navigate into the cloned repository:

cd your-forked-repository

## Making Changes and Committing

Create a New Branch:

git checkout -b new-feature

Make Changes: Modify files or add new files.

## Stage and Commit Changes:

git add .

git commit -m "Added a new feature"

Push Changes to GitHub:

git push origin new-feature

## Creating a Pull Request

* Navigate to Your Fork on GitHub:
* Go to your forked repository on GitHub.
* Open a Pull Request:
* Click the "Compare & pull request" button.
* Ensure the base fork is the original repository and the base branch is the branch you want to merge into (typically main or master).
* The head fork should be your fork, and the compare branch should be the branch you just pushed.
* Submit the Pull Request:
* Add a title and description for your pull request.
* Click "Create pull request".

# Conclusion

Branching allows you to isolate your work, and merging brings those changes together. Forking and pull requests enable you to contribute to other projects seamlessly. By mastering these workflows, you can collaborate more effectively in any Git-based project.