Git – Merge

Last Updated : 16 Mar, 2025

Git is a powerful version control system that helps developers manage code versions efficiently. One of the most fundamental operations in Git is merging, which allows you to integrate changes from one branch into another.

What is Git Merge?

Git merge is a command used to combine the changes from two different branches into one. It helps in integrating work from multiple branches, ensuring that all changes are integrated and up-to-date in the target branch without losing any progress made on either branch.

- **Preserves History:** Keeps commit history of both branches.
- **Automatic and Manual:** Automatically merges unless there are conflicts.
- **Fast-Forward Merge:** Moves the branch pointer forward if no diverging changes exist.
- **Merge Commit:** Creates a special commit to combine histories.
- **No Deletion:** Branches remain intact after merging.
- **Used for Integration:** Commonly integrates feature branches into main branches.

Syntax

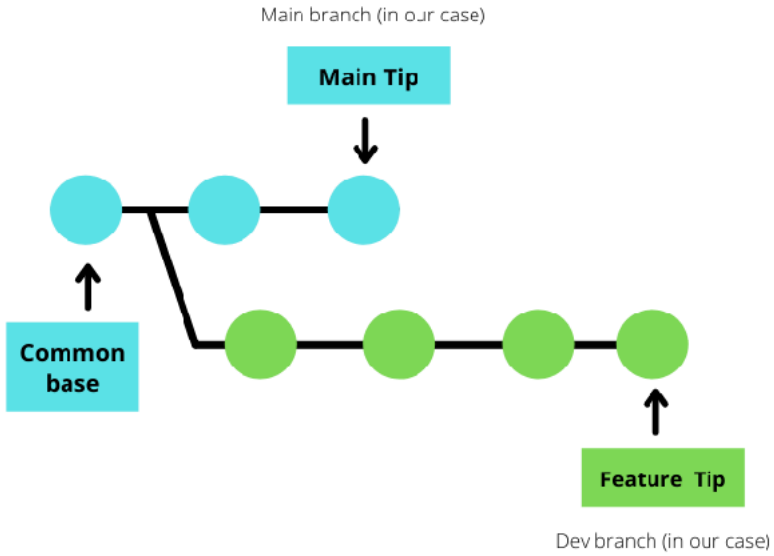
```
git merge <branch-name>
```

How Does Git Merge Work?

The concept of git merging is basically to merge multiple sequences of commits, stored in multiple branches in a unified history, or to be simple you can say in a single branch.

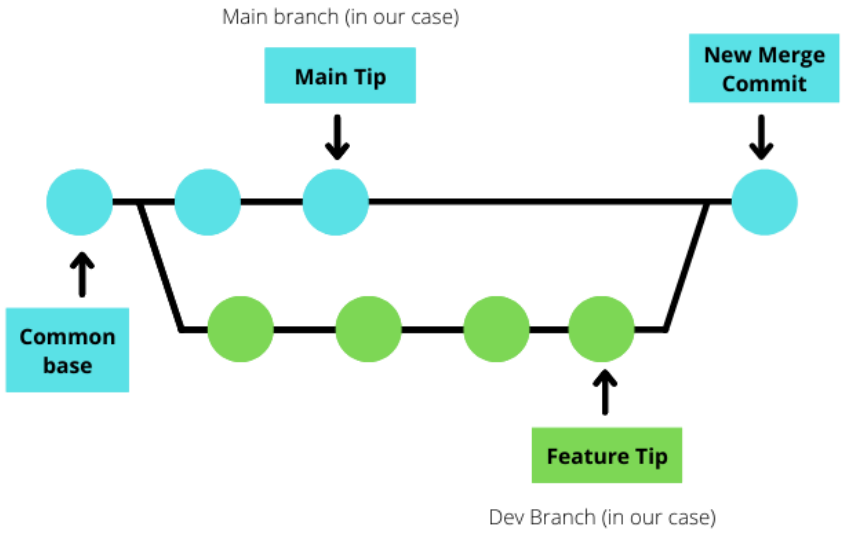
- **Common Base:** When merging two branches, Git looks for the common base commit.

- **Merge Commit:** Once it finds the common base, Git creates a new “merge commit” to combine the changes.
- **Conflict Resolution:** If there are any conflicts between the branches, Git will prompt you to resolve them.



Git – Merge

- In our case, we have two branches one is the default branch called “main” and the other branch named “dev” and this is how our git repo looks before merging.
- Here git finds the common base, creates a new merge commit, and merged them.



Git – Merge

- A git merge operation is performed by running the below command.

- When we perform merging, git always merges with the current branch from where we are performing the operation(in our case it is “main”). By this, the branch being merged is not affected.

```
"git merge <name of the branch to be merged (in our case it is "dev")>".
```

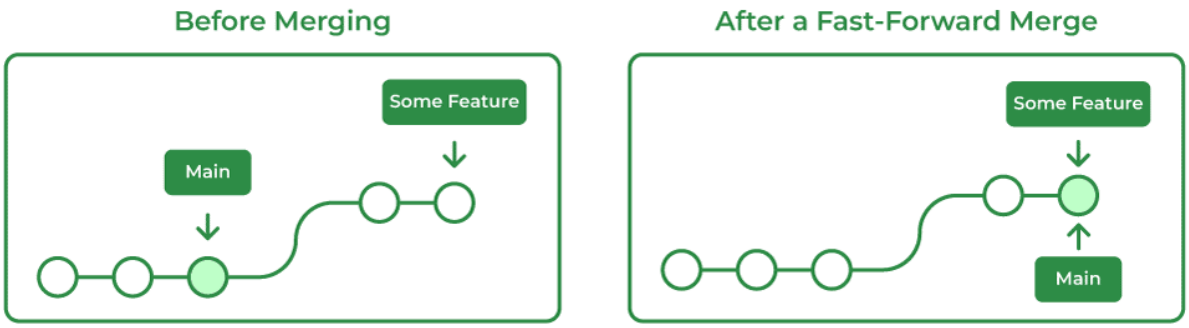
Types of Merging in Git

Git supports several types of merging

In Git, there are two primary types of merging. There are.

1. Fast-forward merging

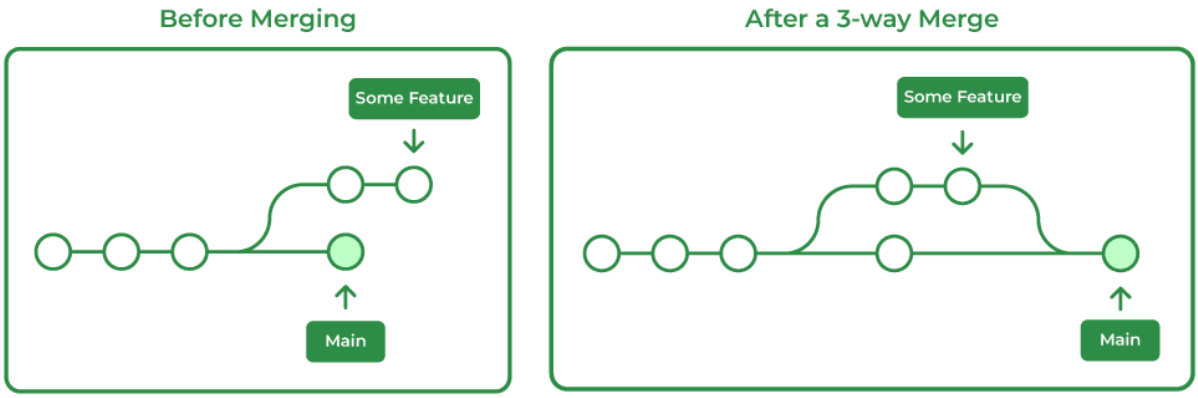
- This occurs when the target branch (e.g., main) is directly ahead of the feature branch (e.g., dev).
- Instead of creating a merge commit, Git simply moves the current branch’s tip to the target branch’s tip.
- Fast-forward merging is only possible when branches haven’t diverged



Git – Merge

2. Three-way merging

- This type occurs when the base branch has changed since the branch was first created.
- Git generates a new merge commit by comparing changes in both branches with the base branch.



Git – Merge

> ***Note :*** *Git also supports some other types of merging like recursive and octopus margin. With the help of a single merge commit "**octopus merging**" can merge multiple branches at once. **"Recursive merging"** is similar to three-way merging but it can handle some more complex merge operations than the three-way merging.*

Essential Commands To Perform Git Merging

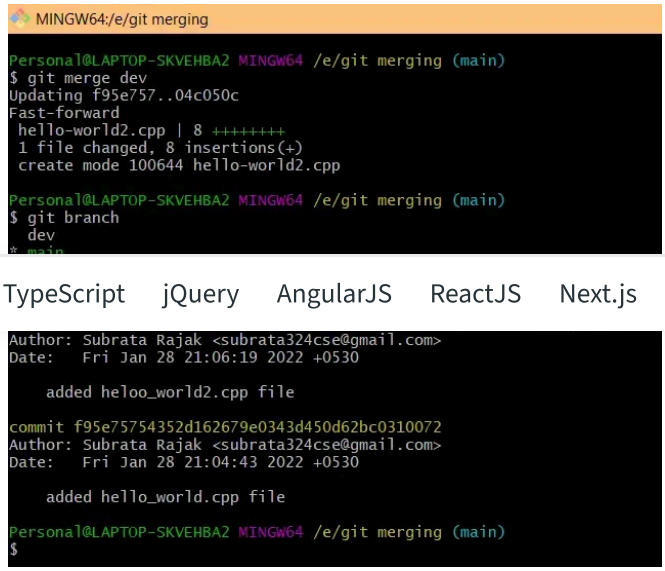
To perform a git merge, you need a Git repository with at least two branches. Here's how to proceed:

- **Create a new branch**

```
git branch <name of the branch you wanna create>
```

- Merge two branches: First, check out the target branch, then run:

```
git merge <name of the current branch>
```



Git – Merge

- Now we have successfully merged our two branches and as you can see we have the same changes or you can say commits in both branches.

Steps To Merge a Branch

To ensure smooth merging, follow these steps:

Step 1: Create a New Branch

Create a new branch from the remote repository you want to merge.

```
git checkout -b <new-branch-name>
```

Step 2: Pull the Latest Changes

Before merging, ensure that you pull the latest changes from both branches (e.g., main and the feature branch).

```
git checkout <target-branch>
git pull origin <target-branch>
git checkout <feature-branch>
git pull origin <feature-branch>
```

Step 3: Merge the Branch

If any conflicts arise, Git will notify you. Resolve them manually before proceeding.

```
git checkout <target-branch>
git merge <feature-branch>
```

Step 4: Test the Merged Code

Make sure the merged code functions correctly by testing it either automatically or manually.

```
# Run tests or manually test your application
```

Step 5: Commit the Merged Code

Once satisfied with the merged code, commit the changes:

```
git commit -m "Merge branch 'dev' into main"
```

Step 6: Push the Merged Branch

Push the changes to the remote repository to share the new merged branch:

```
git push origin main
```

How To Resolve Merge Conflicts?

Git prompts you to resolve conflicts manually when changes are made to the same part of the code in both branches. Here's how to resolve conflicts:

Step 1: Identify the conflict files.

Git will display files that have merge conflicts. These files need manual resolution.

Step 2: Open the conflict files.

Use your preferred editor to open the conflicting files. Look for conflict markers

```
<<<<<<< HEAD
// Code from the current branch
=======
- Code from the merging branch
>>>>>>> branch-name
```

Step 3: Resolve the conflicts.

Remove the unnecessary changes, keeping the most relevant ones.

Step 4: Moving to the staging.

Once resolved, add the files to the staging area:

```
git add <file-name>
```

Step 5: Commit and Push the changes

After resolving conflicts commit the changes by using the below command.Including the message which gives information about changes made while resolving the conflicts.

```
git commit -m "message"
```

Push the changes made to the remote repository by using. Below command.

```
git push
```

Git Merge vs Rebase

Git Merge	Git Rebase
Git merge will create a new commit by combining the changes in one branch with another branch.	Git rebase will integrate one branch commits to another branch by this the commit history will be rewritten.
The commit history of all the branches with which you have merged will be stored.	A new commit will be created when you integrate the changes.
It is useful for incorporating changes from a feature branch or integrating changes from multiple developers.	Rebase is also helpful for resolving conflicts early and maintaining a more streamlined and cohesive commit history.

Uses of Git Merge

- **Incorporate changes:** Integrates changes from another branch into the current one.
- **Consolidate development work:** Merges development done across different branches.
- **Bring feature branches into the main branch:** For example, merging feature branches into main or master.

Next Article

Git Checkout And Merge

Similar Reads

Git Tutorial

Git is an essential tool for developers, enabling them to manage and track changes in their projects. Whether you're working on a solo project or collaborating with a team, Git keeps everything organized and under...

13 min read

Git Introduction

Git Installation and Setup

All Git Commands

Most Used Git Commands

Git Branch

Git Merge

Git - Merge

Git is a powerful version control system that helps developers manage code versions efficiently. One of the most fundamental operations in Git is merging, which allows you to integrate changes from one branch into...

6 min read

Git Checkout And Merge

Git is an essential tool for version control, allowing you to track changes in your code and collaborate with others. Two important Git commands you'll use frequently are git checkout and git merge. Git checkout lets...

6 min read

How to Merge Two Branches in Git?

Version control systems like Git provide powerful tools for managing code changes and collaboration among developers. One common task in Git is merging branches, which allows you to combine the changes made i...

4 min read

How to Merge a Git Branch into Master?

If you're working with Git, merging branches is a fundamental task. Whether you're adding new features, fixing bugs, or improving documentation, merging ensures that your changes are integrated into the main...

3 min read

How to Replace Master Branch with Another Branch in GIT?

In Git, the "master" branch traditionally serves as the primary development branch in many repositories. However, there are situations where you might want to replace the contents of the "master" branch with...

2 min read

Git Merge and Merge Conflict

Let us discuss what merging in git. Merging means combining changes from one branch into another branch. Now let's see how can we perform merging here we can see that we can see one log in the master branch....

2 min read

Git Tools and Integration

Git Remote Repositories

Collaborating with Git

A-143, 7th Floor, Sovereign Corporate Tower, Sector- 136, Noida, Uttar Pradesh (201305)

Registered Address:
K 061, Tower K, Gulshan Vivante Apartment, Sector 137, Noida, Gautam Buddh Nagar, Uttar Pradesh, 201305

Computer Science

Operating Systems
Computer Network
Database Management System
Software Engineering
Digital Logic Design
Engineering Maths
Software Development
Software Testing

DevOps

Git
Linux
AWS
Docker
Kubernetes
Azure
GCP
DevOps Roadmap

System Design

High Level Design
Low Level Design
UML Diagrams
Interview Guide
Design Patterns
OOAD
System Design Bootcamp
Interview Questions

Inteview Preparation

Competitive Programming
Top DS or Algo for CP
Company-Wise Recruitment Process
Company-Wise Preparation
Aptitude Preparation
Puzzles

School Subjects

Mathematics
Physics
Chemistry
Biology
Social Science
English Grammar
Commerce
World GK

GeeksforGeeks Videos

DSA
Python
Java
C++
Web Development
Data Science
CS Subjects