### **Basic Git Questions**

### **1.**What is Git and why is it used?

**Git** is a distributed version control system (DVCS) that allows developers to track changes in their codebase, collaborate with others, and manage different versions of their projects efficiently.

**why is it used:**

Git is a useful tool that helps to manage changes in code or any documents, It generally allows collaboration between developers and writers and enables them easy version control.

2**.Explain the difference between Git and GitHub.**

|  |  |
| --- | --- |
| **Git** | **GitHub** |
| Git is a software. | GitHub is a service. |
| Git is a command-line tool | GitHub is a graphical user interface |
| Git is installed locally on the system | GitHub is hosted on the web |
| Git is maintained by linux. | GitHub is maintained by Microsoft. |
| Git is focused on version control and code sharing. | GitHub is focused on centralized source code hosting. |
| Git is a version control system to manage source code history. | GitHub is a hosting service for Git repositories. |

**3.How do you install Git on your machine?**

1. Navigate to the latest [Git for Windows installer](https://gitforwindows.org/) and download the latest version.
2. Once the installer has started, follow the instructions as provided in the **Git Setup** wizard screen until the installation is complete.
3. Open the windows command prompt (or **Git Bash** if you selected not to use the standard Git Windows Command Prompt during the Git installation).
4. Type git version to verify Git was installed.

**4.How do you configure your username and email in Git?**

**Username**: git config --global user.name "Manu"

**Email Address**: git config --global user.email “[manu@gmail.com](mailto:manu@gmail.com)”

**5. What is a repository in Git?**

In Git, a repository (often abbreviated as "repo") is a storage location where a collection of files and their revision history are kept. It serves as a central hub for a project, enabling collaboration among developers by allowing them to access, share, and modify the project's source code.

**6.How do you create a new Git repository?**

To create a new repo, you'll use the git init command. git init is a one-time command you use during the initial setup of a new repo. Executing this command will create a new git subdirectory in your current working directory.

**7. How do you clone a repository from GitHub?**

Go to the GitHub repository page.

Look for "Code" button on the right side of the page. Click on it and copy the URL that appears. This URL will be something like.

use the git clone command followed by the URL you copied

* git clone git@github.com:kethireddy123/git.manu.git

**8.What is the purpose of the .gitignore file?**

The purpose of gitignore files is to ensure that certain files not tracked by Git remain untracked. To stop tracking a file that is currently tracked, use git rm --cached to remove the file from the index.

**9.How do you check the status of your working directory in Git?**

The git status command displays the state of the working directory and the staging area. It lets you see which changes have been staged, which haven't, and which files aren't being tracked by Git. Status output does not show you any information regarding the committed project history.

* **git status**

**10.How do you add files to the staging area in Git?**

* git add [filename]

**11.Explain the concept of commits in Git.**

In Git, a commit represents a snapshot of your project at a particular point in time. When you make changes to your project files, you stage those changes using the git add command, and then you commit those changes using the git commit command.

**12.How do you create a new commit in Git?**

* **git commit -m "commit message"**

**13. What is the purpose of the git log command?**

The git log command displays all of the commits in a repository's history. By default, the command displays each commit's: Secure Hash Algorithm (SHA).

**14. How do you view the history of commits in a repository?**

* **git log**

**15.How do you view the changes made in a commit?**

* **git show**

**16.What is branching in Git and why is it useful?**

Branches allow you to work on different parts of a project without impacting the main branch. When the work is complete, a branch can be merged with the main project. You can even switch between branches and work on different projects without them interfering with each other.

**17.How do you create a new branch in Git?**

* **git branch branch-name**

**18.How do you switch between branches in Git?**

* **git checkout branch-name**
* **git switch branch-name**

**19.What is the difference between git merge and git rebase?**

| **Git Merge** | **Git Rebase** |
| --- | --- |
| Git Merge merges two branches to create a “feature” branch. | Git Rebase rebases the feature branch to add the feature branch to the main branch. |
| Git Merge is comparatively easy. | Git Rebase is comparatively harder. |
| Git Merge safeguards history. | Git Rabse doesn’t safeguard history. |
| Git Merge is more suitable for projects with the less active main branch. | Git Rebase is suitable for projects with frequently active main branches. |
| Git Merge forms a chain-like structure. | Git Rebase forms a linear structure. |
| Git Merge is preferable for large no. of people working on a project. | Git Rebase is preferable for small groups of people. |
| Single line command is:  *git merge feature main* | Single line command is:  *git rebase main* |

**20.How do you resolve merge conflicts in Git?**

Step 1: The easiest way to resolve a conflicted file is to open it and make any necessary changes.

Step 2: After editing the file, we can use the git add a command to stage the new merged content.

Step 3: The final step is to create a new commit with the help of the git commit command.

Step 4: Git will create a new merge commit to finalize the merge.

How to Resolve Merge Conflicts in Git?

Lesson 7 of 12By Sana Afreen

Last updated on Oct 11, 20231256294

How to Resolve Merge Conflicts in Git?

PreviousNext

Table of Contents

What is a Git Merge Conflict?How to Resolve Merge Conflicts in Git?Git Commands to Resolve ConflictsTypes of Git Merge Conflicts Demo: Resolving Git Merge ConflictsView More

Git is one of the most popular source-control systems that enable software development professionals in all industries, enabling multiple team members to work concurrently on projects. Since many users are simultaneously working from different places on the same file, however, you may end up with a merge conflict. This article explains the basics of Git merge conflicts and one of the advanced operations of Git: resolving a Git merge conflict.

What is a Git Merge Conflict?

A git merge conflict is an event that takes place when Git is unable to automatically resolve differences in code between two commits. Git can merge the changes automatically only if the commits are on different lines or branches.

The following is an example of how a Git merge conflict works:

pull-push.

Let’s assume there are two developers: Developer A and Developer B. Both of them pull the same code file from the remote repository and try to make various amendments in that file. After making the changes, Developer A pushes the file back to the remote repository from his local repository. Now, when Developer B tries to push that file after making the changes from his end, he is unable to do so, as the file has already been changed in the remote repository.

To prevent such conflicts, developers work in separate isolated branches. The Git merge command combines separate branches and resolves any conflicting edits.

Now that we have gone through the basics of the Git merge conflict, let’s look at the various types of conflicts next.

Get All Your Career Growth Questions Answered!

DevOps Engineer Masters ProgramEXPLORE PROGRAMGet All Your Career Growth Questions Answered!

How to Resolve Merge Conflicts in Git?

There are a few steps that could reduce the steps needed to resolve merge conflicts in Git.

Step 1: The easiest way to resolve a conflicted file is to open it and make any necessary changes.

Step 2: After editing the file, we can use the git add a command to stage the new merged content.

Step 3: The final step is to create a new commit with the help of the git commit command.

Step 4: Git will create a new merge commit to finalize the merge.

Git Commands to Resolve Conflicts

1. git log --merge

The git log --merge command helps to produce the list of commits that are causing the conflict.

2. git diff

The git diff command helps to identify the differences between the states repositories or files.

3. git checkout

The git checkout command is used to undo the changes made to the file, or for changing branches.

4. git reset --mixed

The git reset --mixed command is used to undo changes to the working directory and staging area.

5. git merge --abort

The git merge --abort command helps in exiting the merge process and returning back to the state before the merging began.

6. git reset

The git reset command is used at the time of merge conflict to reset the conflicted files to their original state

**Advanced Git Questions**

**21. What is the purpose of the git stash command?**

The git stash command is used to temporarily store changes in your working directory that are not yet ready to be committed. It's useful in scenarios where you need to switch to a different branch, pull changes from a remote repository, or perform other tasks without committing your current changes. Stashing allows you to save your work in progress and retrieve it later when needed. It's particularly helpful when you're in the middle of working on something but need to switch focus temporarily. Once you've stashed your changes, you can continue with your other tasks, and later on, you can reapply your stashed changes back into your working directory. This command is an essential tool for managing work in Git repositories and helps keep your workflow organized and flexible.

**22. How do you apply stashed changes in Git?**

* **git stash apply**

**23. What is the purpose of the git tag command?**

The git tag command in Git is used to mark specific points in the commit history as important, such as release points or version numbers. It allows you to create, list, delete, and verify tags in a repository. Tags serve as references to specific commits and are often used to denote stable releases of software.

**24. How do you create and push tags to a remote repository?**

* **Git tag v1.o**
* **Git push origin v1.o**

**25. Explain the concept of remote repositories in Git.**

In Git, a remote repository is a version of your project that is hosted on another server, separate from your local machine. This setup allows multiple developers to collaborate on the same project by pushing and pulling changes to and from the remote repository.

**26. How do you add a remote repository in Git?**

**Git remote add origin** [**git@github.com:kethireddy123/git.manu.git**](mailto:git@github.com:kethireddy123/git.manu.git)

**27. How do you push changes to a remote repository?**

* **Git push -u origin branch name**

**28. How do you pull changes from a remote repository?**

* **git pull origin branch\_name**

**29. What is the purpose of the git fetch command?**

git fetch is a primary command used to download contents from a remote repository. git fetch is used in conjunction with git remote , git branch , git checkout , and git reset to update a local repository to the state of a remote.

**30. How do you delete a branch in Git?**

* **git branch -d branchname**

**Git Exercises**

**31. Create a new Git repository and configure your username and email.**

**Username**: git config --global user.name "Manu"

**Email Address**: git config --global user.email “[manu@gmail.com](mailto:manu@gmail.com)”

**git config --global --list**

**user.name=manu**

[**user.mail=hreddy9908@gmail.com**](mailto:user.mail=hreddy9908@gmail.com)

**32. Create a file, add some content to it, and commit the changes.**

**PS C:\Users\LENOVO V330\Desktop\manu> git init**

**Reinitialized existing Git repository in C:/Users/LENOVO V330/Desktop/manu/.git/**

**PS C:\Users\LENOVO V330\Desktop\manu> git add newfile.html**

**PS C:\Users\LENOVO V330\Desktop\manu> git commit -m "commit newfile"**

**[manu a92aef2] commit newfile**

**1 file changed, 11 insertions(+)**

**create mode 100644 newfile.html**

**PS C:\Users\LENOVO V330\Desktop\manu>**

**33. Create a .gitignore file and add rules to ignore specific files and directories.**

**PS C:\Users\LENOVO V330\Desktop\manu> git init**

**Reinitialized existing Git repository in C:/Users/LENOVO V330/Desktop/manu/.git/**

**PS C:\Users\LENOVO V330\Desktop\manu> git add .gitignore**

**34. Clone an existing repository from GitHub and make some changes.**

**PS C:\Users\LENOVO V330\Desktop\manu> git clone git@github.com:GreemusDevOps/DevopsprojecCloning into 'Devopsprojects'...**

**remote: Counting objects: 100% (168/168), done.**

**remote: Compressing objects: 100% (104/104), done.**

**remote: Total 168 (delta 59), reused 151 (delta 47), pack-reused 0**

**Receiving objects: 100% (168/168), 18.69 KiB | 83.00 KiB/s, done.**

**Resolving deltas: 100% (59/59), done.**

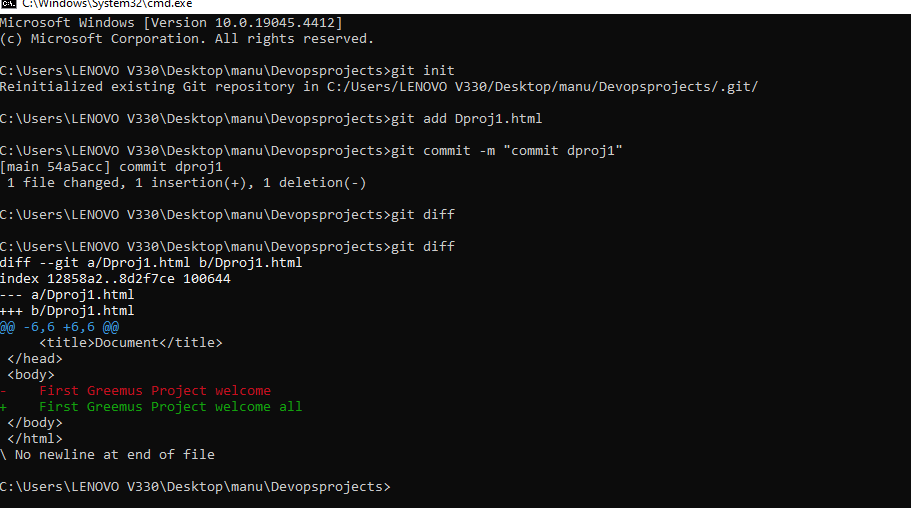
**PS C:\Users\LENOVO V330\Desktop\manu> git init**

**Reinitialized existing Git repository in C:/Users/LENOVO V330/Desktop/manu/.git/**

**PS C:\Users\LENOVO V330\Desktop\manu> git add Dproj1.html**

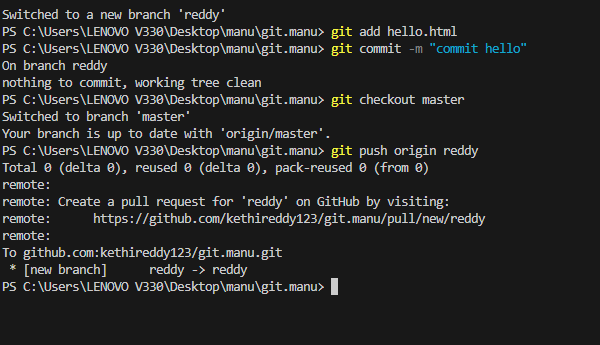
**fatal: pathspec 'Dproj1.html' did not match any files**

**PS C:\Users\LENOVO V330\Desktop\manu> git add "Devopsprojects"**

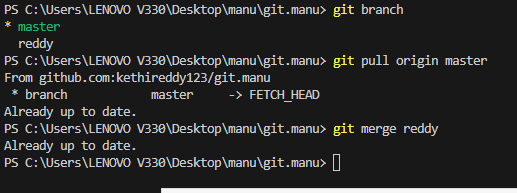


**35. Create a new branch, make some changes, and switch back to**

**the main branch.**

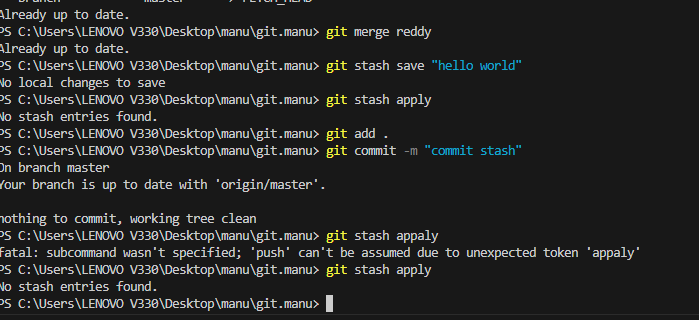


**36. Merge changes from a feature branch into the main branch.**

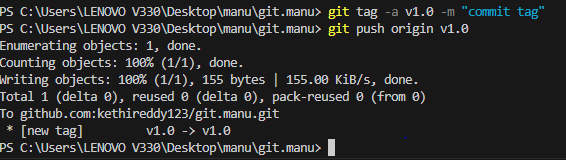


**37. Resolve a merge conflict between two branches.**

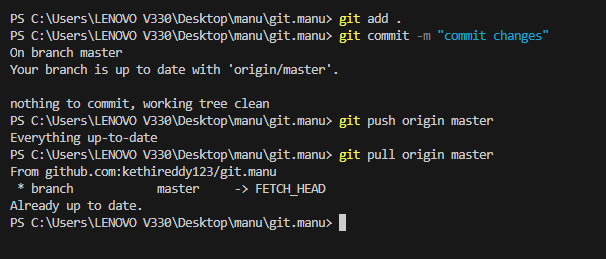
**38. Use git stash to save your work and then apply the stashed changes.**



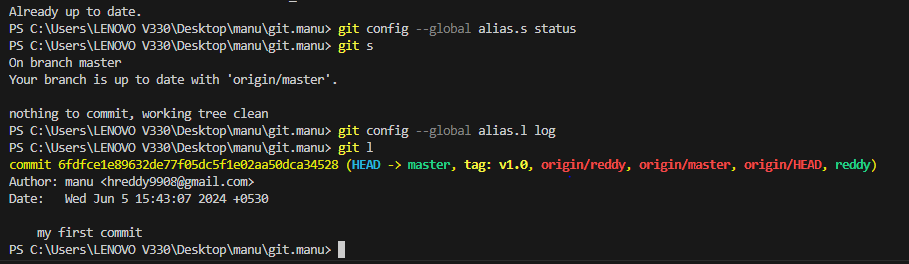
**39. Create a tag for a specific commit and push the tag to the remote repository.**



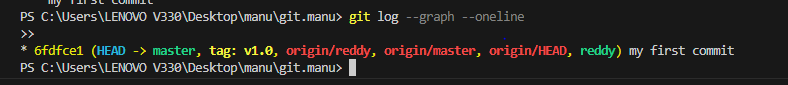
**40. Add a remote repository, push your local changes, and pull changes from the remote repository.**



**41.Create a Git alias for a commonly used command.**



**42.View the commit history with a graph representation.**



**43.Revert a commit and understand the differences between git revert and git reset.**

Git revert is a command used to create a new commit that undoes the changes introduced by a specific commit. Unlike git reset, which modifies the project history, git revert creates a new commit that effectively cancels out the changes made in the target commit. This is a safe way to undo changes without altering the commit history.

44.Squash multiple commits into a single commit using git rebase.

Step 1: Identify the commits to squash

Step 2: Initiate an interactive rebase

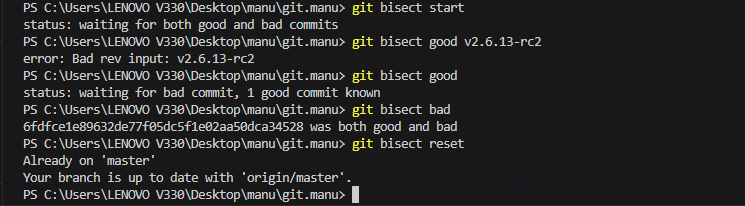
Step 3: Specify the squash action.

Step 4: Edit the commit message

Step 5: Complete the rebase.

Step 6: Push the changes.

**45.Use git bisect to find the commit that introduced a bug.**



46.Set up a Git hook to run a script before or after certain Git events.

Set up a Git hook to run a script before or after certain Git events.

To install the hook, you can either create a symlink to it in . git/hooks , or you can simply copy and paste it into the . git/hooks directory whenever the hook is updated. As an alternative, Git also provides a Template Directory mechanism that makes it easier to install hooks automatically.

47.Use git cherry-pick to apply a commit from one branch to another.

Using Git Cherry-Pick To Apply A Commit From Another Branch

1. Identify The Commit You Want To Cherry-pick.

2. Switch To The Target Branch.

3. Using The Cherry-pick Command.

4. Resolve Any Conflicts That Arise.

5. Commit The Changes.

6. Push The Changes.

**48.Learn how to use git blame to find out who made specific changes to a file.**

git blame only operates on individual files. A file-path is required for any useful output. The default execution of git blame will simply output the commands help menu. For this example, we will operate on the README.MD file. It is a common open source software practice to include a README file in the root of a git repository as documentation source for the project.

**49.Configure and use a Git GUI client**.

git config --global user.name "manu"

git config --global user.email "hreddy9908@gmail.com"

**50.Create a fork of a repository on GitHub, make changes, and open a pull request.**

