How to Resolve Merge Conflicts in Git?

[Git](https://www.simplilearn.com/tutorials/git-tutorial/what-is-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

What is Git?

Git is an open-source, distributed version control system (VCS), which has a remote repository on the server-side and a local repository on the client-side. This means that the file or code is not present in a central server, but there is a copy of the file stored on the client’s computer.

A distributed version control system enables multiple developers to work in parallel with each other without any code conflicts. Git enables developers to revert and go back to an older version of the code whenever necessary.

Git helps both developers and non-tech professionals by keeping track of their project files. It makes it easier for multiple individuals to work together, and it plays an extremely significant role in big projects that involve large teams.

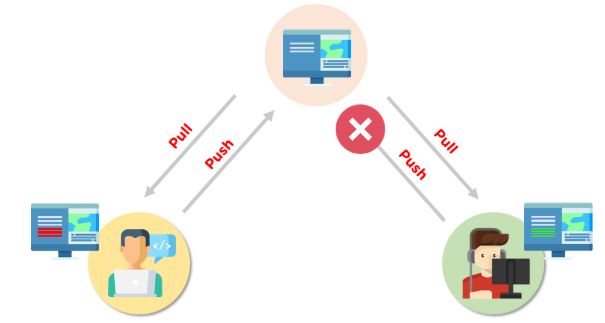
Basic Git Commands

* git config
* git init
* git add
* git diff
* git commit
* git reset
* git status
* git merge
* [git push](https://www.simplilearn.com/tutorials/git-tutorial/git-push-command)
* [git pull](https://www.simplilearn.com/tutorials/git-tutorial/git-pull-request)

What is a Git Merge Conflict?

A 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:



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.

Choose the Best for Your Career!

Caltech Program in DevOps[EXPLORE PROGRAM](https://www.simplilearn.com/pgp-devops-certification-training-course?source=GhPreviewCTABanner)

Choose the Best for Your Career!

Types of Git Merge Conflicts

There are two points when a merge can enter a conflicted state:

1. Starting the Merge Process

If there are changes in the working directory’s stage area for the current project, merging won’t start.

In this case, conflicts happen due to pending changes that need to be stabilized using different [Git commands](https://www.simplilearn.com/tutorials/git-tutorial/git-commands).

2. During the Merge Process

The failure during the merge process indicates that there is a conflict between the local branch and the branch being merged.

In this case, Git resolves as much as possible, but there are things that have to be resolved manually in the conflicted files.

We will now go over resolving merge conflicts in Git.

How to Resolve Merge Conflicts in Git?

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

1. The easiest way to resolve a conflicted file is to open it and make any necessary changes
2. After editing the file, we can use the git add a command to stage the new merged content
3. The final step is to create a new commit with the help of the git commit command
4. Git will create a new merge commit to finalize the merge

Let us now look into the Git commands that may play a significant role in resolving conflicts.

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

FREE DevOps Certification Training

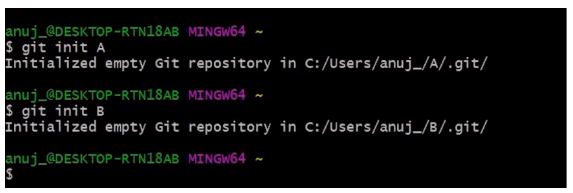
Master the fundamentals of DevOps[START LEARNING](https://www.simplilearn.com/learn-devops-basics-skillup?utm_source=frs&utm_medium=skillup-course-banner&utm_campaign=frs-tutorial-skillup-course-promotion)

FREE DevOps Certification Training

Demo: Resolving Git Merge Conflicts

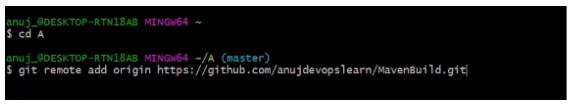
First, initialize two repositories:

|  |
| --- |
| git init A  git init B |



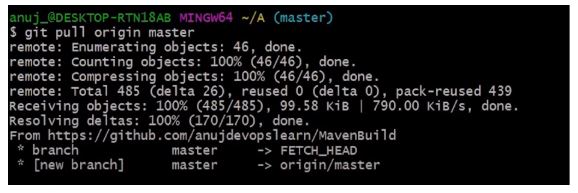
Add the remote address in the A repository:

|  |
| --- |
| git remote add origin \*address\* |



The next step is to pull all the changes in the central repository to the local repository.

|  |
| --- |
| git pull origin master |



Follow the same process to add the origin in the B repository.

|  |
| --- |
| git remote add origin \*address\* |

the-pull

The pull command is executed again to retrieve all the content from the remote repository and move it to the local repository.

|  |
| --- |
| git pull origin master |



Both of these repositories represent two different repositories of two different developers.

Choose the Best for Your Career!

Caltech Program in DevOps[EXPLORE PROGRAM](https://www.simplilearn.com/pgp-devops-certification-training-course?source=GhPreviewCTABanner)

Choose the Best for Your Career!

Let's get back to the A repository.

|  |
| --- |
| cd ../A |



In the A repository, a readme file is opened in order to make various changes.

|  |
| --- |
| vi README.md |



Make the necessary changes in the file, and then save it.

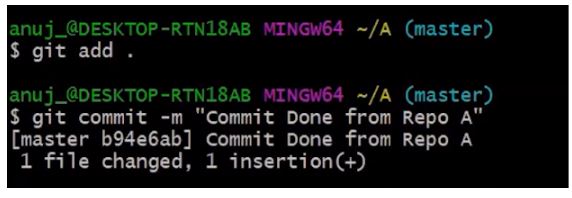
The git status command is then executed in order to see the reflected changes.

|  |
| --- |
| git status |



The next step is to add these changes to the staging area and commit them.

|  |
| --- |
| git add.  git commit -m \*commit message\* |



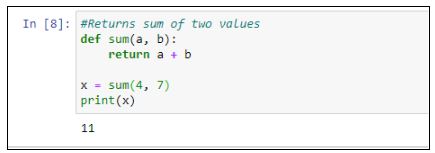
Learn from Experienced Industry Experts!

DevOps Engineer Masters Program[EXPLORE PROGRAM](https://www.simplilearn.com/devops-engineer-masters-program-certification-training?source=GhPreviewCTABanner)

Learn from Experienced Industry Experts!

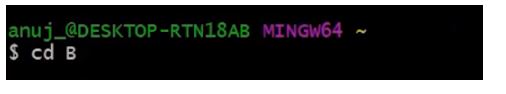
After the commit is finished, the changed file is pushed to the remote repository.

|  |
| --- |
| git push origin master |



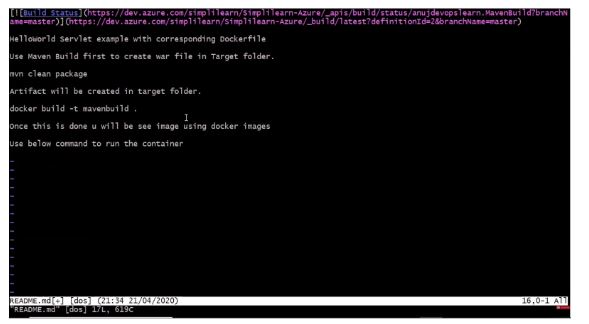
Now, return to the B repository.

|  |
| --- |
| cd B |



Open a readme file

|  |
| --- |
| vi README.md |



FREE GIT Training

Learn the basics of GIT[ENROLL NOW](https://www.simplilearn.com/learn-git-basics-skillup?utm_source=frs&utm_medium=skillup-course-banner&utm_campaign=frs-tutorial-skillup-course-promotion)

FREE GIT Training

Make changes to the file, save it, and close it. After that, add the changed file and commit it.

|  |
| --- |
| git add.  git commit -m \*commit message\* |



The next step is to push the file to the remote repository.

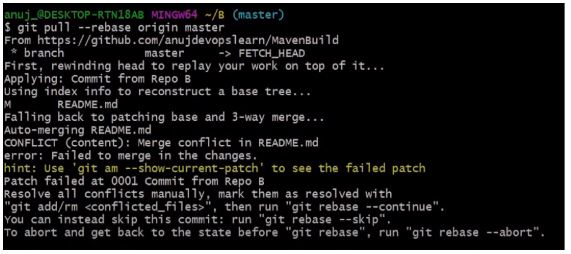
|  |
| --- |
| git push |



An error is shown, meaning that the updates are rejected.

Next, we need to execute:

|  |
| --- |
| git -- rebase origin master |

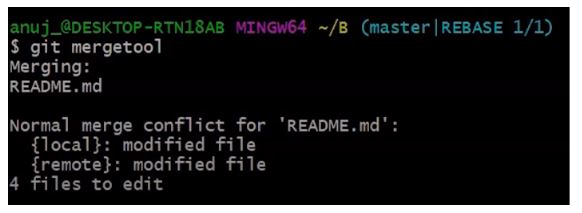


Currently, there are visible conflicts that need to be resolved manually.

If you want to skip this commit, you can type git rebase --skip, or if you want to abort this rebase, you can type git rebase --abort.

After managing this conflict manually, we will open the merge tool.

|  |
| --- |
| git mergetool |



After we input this command, all of the files will be processed.



These are all the processes and the modifications done in the file.

You can see three different files there, and you can see everything that was added or removed.

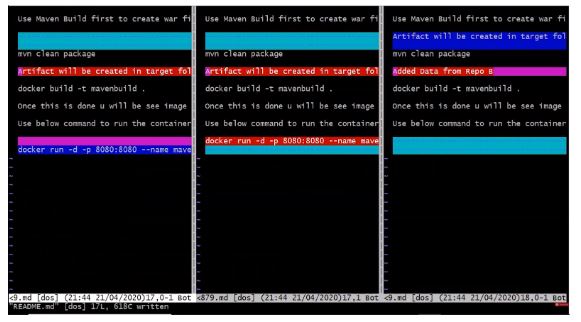
After scrolling, you can verify where exactly the conflict happened.



You can then decide if you want to continue with this particular file or not. I will proceed with removing that line.

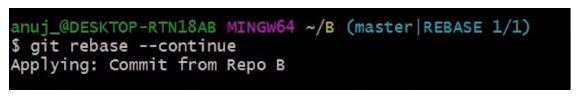


Manual modifications have allowed us to resolve file conflicts. Save the file and close the final file.

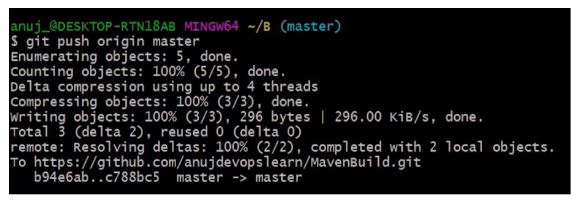


Next, we will run:

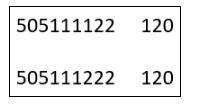
|  |
| --- |
| git rebase --continue |



Now, when the conflict is resolved, we must be able to push the file to the remote repository.



You can now check the commits in your remote repository.



Explore the opportunities of working with the latest DevOps tools such as Docker, Git, Jenkins, and more by choosing our DevOps Engineer Certification Course. Grab your seat fast by contacting our admission counselor TODAY!

prevNext