

Git Rebase Workflow

A. Purpose:

All team members must refer to this document before modifying any local project files.

This document walks the user through a Git rebase workflow using topic branching. The instructions explain how to:

- Create a topic branch.
- Apply codes modifications to the branch.
- Update the branch with changes from the parent branch.
- Resolve rebase conflicts.
- Merge the branch back into the parent branch.

By following this document we will reduce the frequency and complexity of merge conflicts, which will save our team a considerable amount of time.

B. Required Materials

- A remote Git repository
- A computer with command-line Git installed and configured
- A local clone of a remote Git repository with at least one remote-tracking branch

C. Procedure

a. Create Topic Branch

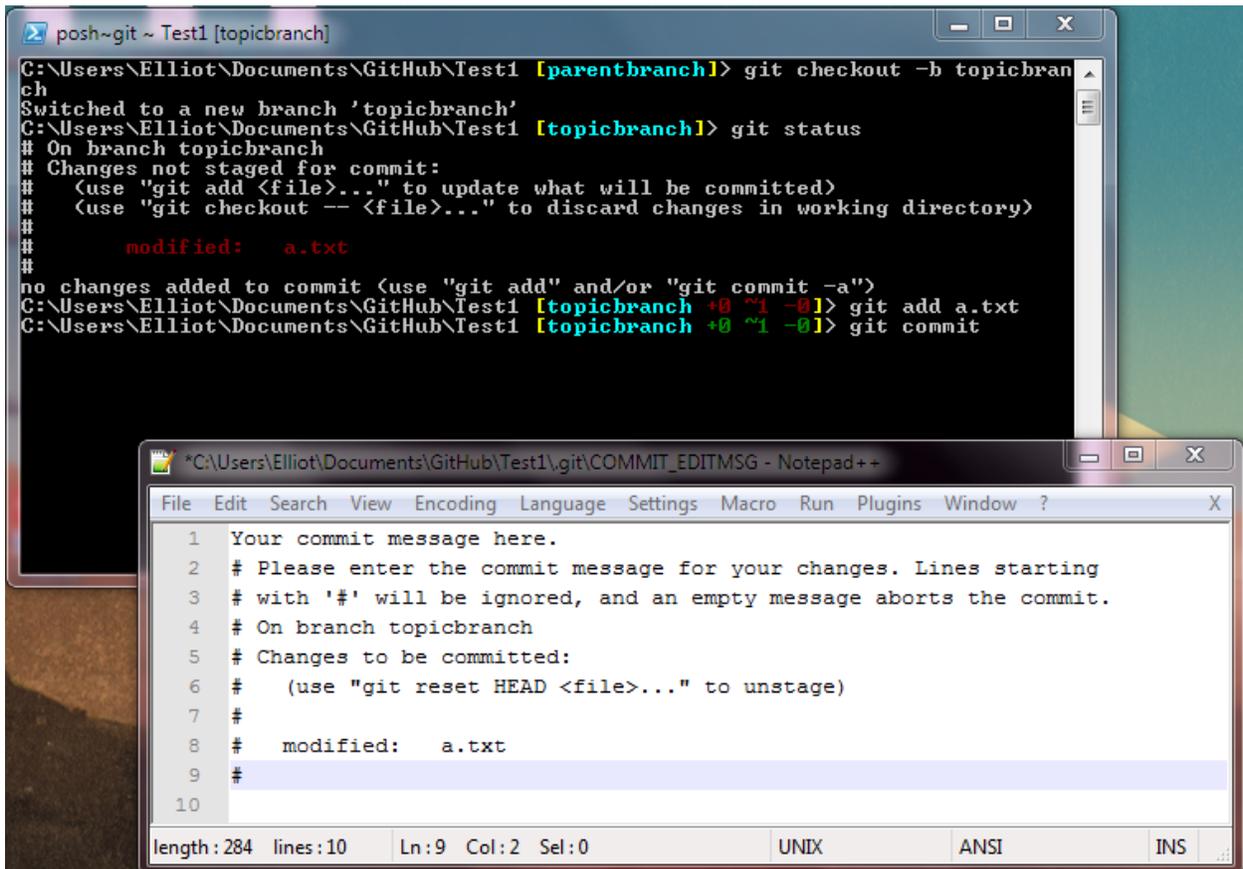
1. Checkout the parent branch and update it.
 - a. `git checkout parentbranch`
 - b. `git pull`
2. Create and checkout a local topic branch
 - a. `git checkout -b topicbranch`

b. Commit Code Modifications

1. Create, delete, and modify files. Proceed to step 2 when you are ready to commit.

Note: Use `git status` to view modified files.

2. Stage changes you want to commit.
 - a. `git add a.txt`
3. Create commit.
 - a. `git commit`
 - b. Git should automatically launch your text editor.
 - c. Write a commit message
 - d. Save the file and close the text editor.



The image shows a terminal window and a Notepad++ editor. The terminal window displays the following commands and output:

```
posh~git ~ Test1 [topicbranch]
C:\Users\Elliott\Documents\GitHub\Test1 [parentbranch]> git checkout -b topicbranch
Switched to a new branch 'topicbranch'
C:\Users\Elliott\Documents\GitHub\Test1 [topicbranch]> git status
# On branch topicbranch
# Changes not staged for commit:
#   (use "git add <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working directory)
#
#       modified:   a.txt
#
no changes added to commit (use "git add" and/or "git commit -a")
C:\Users\Elliott\Documents\GitHub\Test1 [topicbranch +0 ~1 -0]> git add a.txt
C:\Users\Elliott\Documents\GitHub\Test1 [topicbranch +0 ~1 -0]> git commit
```

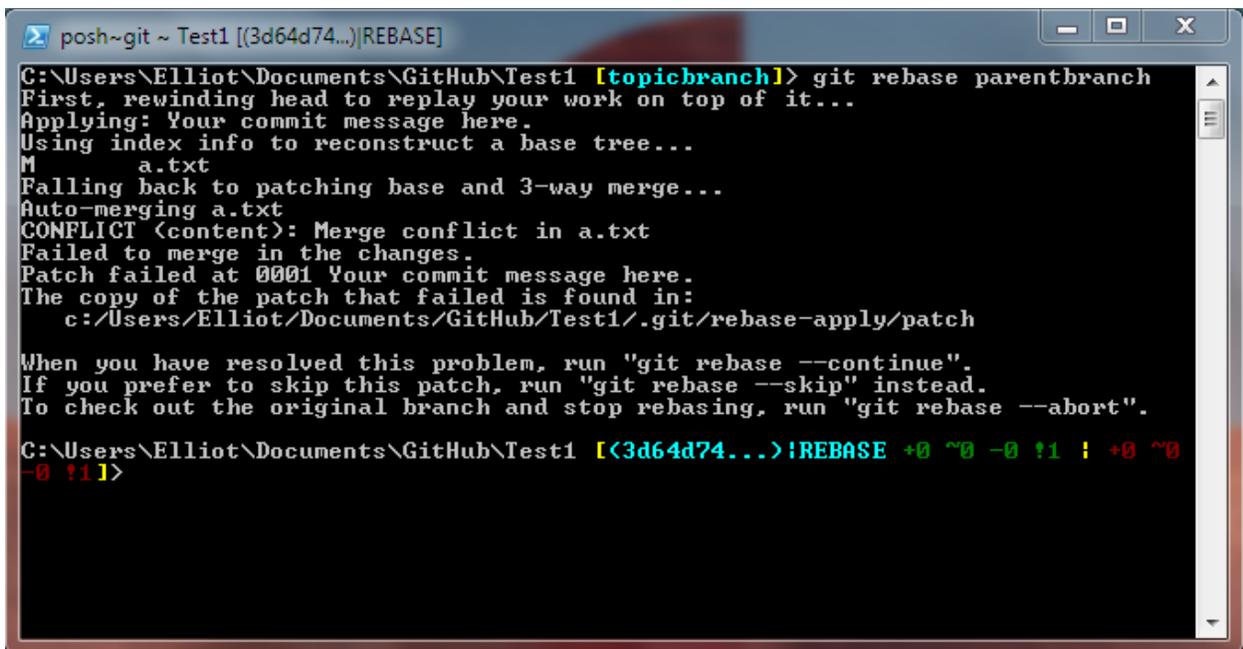
The Notepad++ editor shows the commit message template:

```
*C:\Users\Elliott\Documents\GitHub\Test1\.git\COMMIT_EDITMSG - Notepad++
File Edit Search View Encoding Language Settings Macro Run Plugins Window ?
1 Your commit message here.
2 # Please enter the commit message for your changes. Lines starting
3 # with '#' will be ignored, and an empty message aborts the commit.
4 # On branch topicbranch
5 # Changes to be committed:
6 #   (use "git reset HEAD <file>..." to unstage)
7 #
8 #   modified:   a.txt
9 #
10
length: 284 lines: 10 Ln: 9 Col: 2 Sel: 0 UNIX ANSI INS
```

Git should automatically launch a text editor

c. Rebase onto Parent Branch

1. Update the parent branch with remote changes.
 - a. `git checkout parentbranch`
 - b. `git pull`
2. Rebase the topic branch
 - a. `git checkout topicbranch`
 - b. `git rebase parentbranch`
3. If there are rebase conflicts continue to *d. Resolve Rebase Conflicts*
4. If there are no rebase conflicts, skip to *e.*



```
posh~git ~ Test1 [(3d64d74...)|REBASE]
C:\Users\Elliot\Documents\GitHub\Test1 [topicbranch]> git rebase parentbranch
First, rewinding head to replay your work on top of it...
Applying: Your commit message here.
Using index info to reconstruct a base tree...
M   a.txt
Falling back to patching base and 3-way merge...
Auto-merging a.txt
CONFLICT (content): Merge conflict in a.txt
Failed to merge in the changes.
Patch failed at 0001 Your commit message here.
The copy of the patch that failed is found in:
  c:/Users/Elliot/Documents/GitHub/Test1/.git/rebase-apply/patch

When you have resolved this problem, run "git rebase --continue".
If you prefer to skip this patch, run "git rebase --skip" instead.
To check out the original branch and stop rebasing, run "git rebase --abort".

C:\Users\Elliot\Documents\GitHub\Test1 [(3d64d74...)|REBASE +0 ~0 -0 ?1 ! +0 ~0
-0 ?1!]>
```

Rebase conflicts—failed to auto-merge

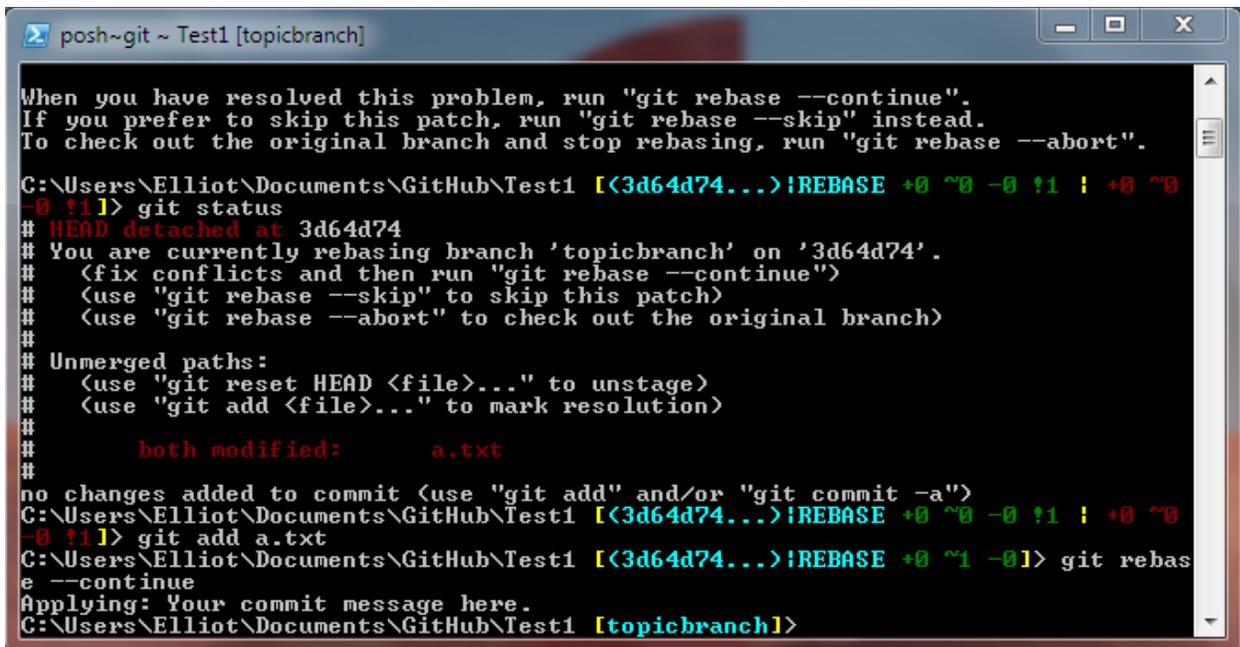
d. Resolve Rebase Conflicts

- a. Use `git status` to determine unmerged files.
- b. Resolve unmerged files.
 - a. If you have a mergetool set up, run the mergetool
 - i. `git mergetool`
 - ii. The mergetool will open each unmerged file one at a time.
When you have resolved the file, save and exit the mergetool.
Git will automatically stage the fixed files.
 - b. If you do not have a mergetool set up, resolve the files manually.
 - i. Open an unmerged file in your text editor.
 - ii. Resolve the conflict and save your changes.
 - iii. Stage the resolved file.
 1. `git add a.txt`

Note: If you incorrectly resolve a file you can revert it to its unmerged state with `git checkout -m a.txt`.

Note: You can cancel the rebase at any time with `git rebase --abort`.

- c. Complete the rebase.
 - a. `git rebase --continue`



```
posh~git ~ Test1 [topicbranch]
When you have resolved this problem, run "git rebase --continue".
If you prefer to skip this patch, run "git rebase --skip" instead.
To check out the original branch and stop rebasing, run "git rebase --abort".
C:\Users\Elliot\Documents\GitHub\Test1 [(3d64d74...)!REBASE +0 ~0 -0 !1 ! +0 ~0
-0 !1]> git status
# HEAD detached at 3d64d74
# You are currently rebasing branch 'topicbranch' on '3d64d74'.
# (fix conflicts and then run "git rebase --continue")
# (use "git rebase --skip" to skip this patch)
# (use "git rebase --abort" to check out the original branch)
#
# Unmerged paths:
# (use "git reset HEAD <file>..." to unstage)
# (use "git add <file>..." to mark resolution)
#
#       both modified:   a.txt
#
no changes added to commit (use "git add" and/or "git commit -a")
C:\Users\Elliot\Documents\GitHub\Test1 [(3d64d74...)!REBASE +0 ~0 -0 !1 ! +0 ~0
-0 !1]> git add a.txt
C:\Users\Elliot\Documents\GitHub\Test1 [(3d64d74...)!REBASE +0 ~1 -0]> git rebase
--continue
Applying: Your commit message here.
C:\Users\Elliot\Documents\GitHub\Test1 [topicbranch]>
```

A resolved rebase conflict

e. Repeat steps *b. Commit Code Modifications*, *c. Rebase onto Parent Branch*, and *d. Resolve Rebase Conflicts* until feature is fully implemented.

f. Merge Topic Branch into Parent Branch

1. Update the parent branch and rebase onto it.
 - a. `git checkout parentbranch`
 - b. `git pull`
 - c. `git checkout topicbranch`
 - d. `git rebase parentbranch`
 - e. If there are rebase conflicts, refer to *d. Resolve Rebase Conflicts*.
2. Checkout the parent branch.
 - a. `git checkout parentbranch`
3. Merge the topic branch into the parent branch.
 - a. `git merge topicbranch`

Note: The merge should be a fast-forward. A merge conflict or a message like “Merge made by the ‘recursive’ strategy” indicates that you did not rebase the topic branch.

4. Push commits on the parent branch to the remote.
 - a. `git push`
5. Delete the topic branch.
 - a. `git branch -d topicbranch`