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

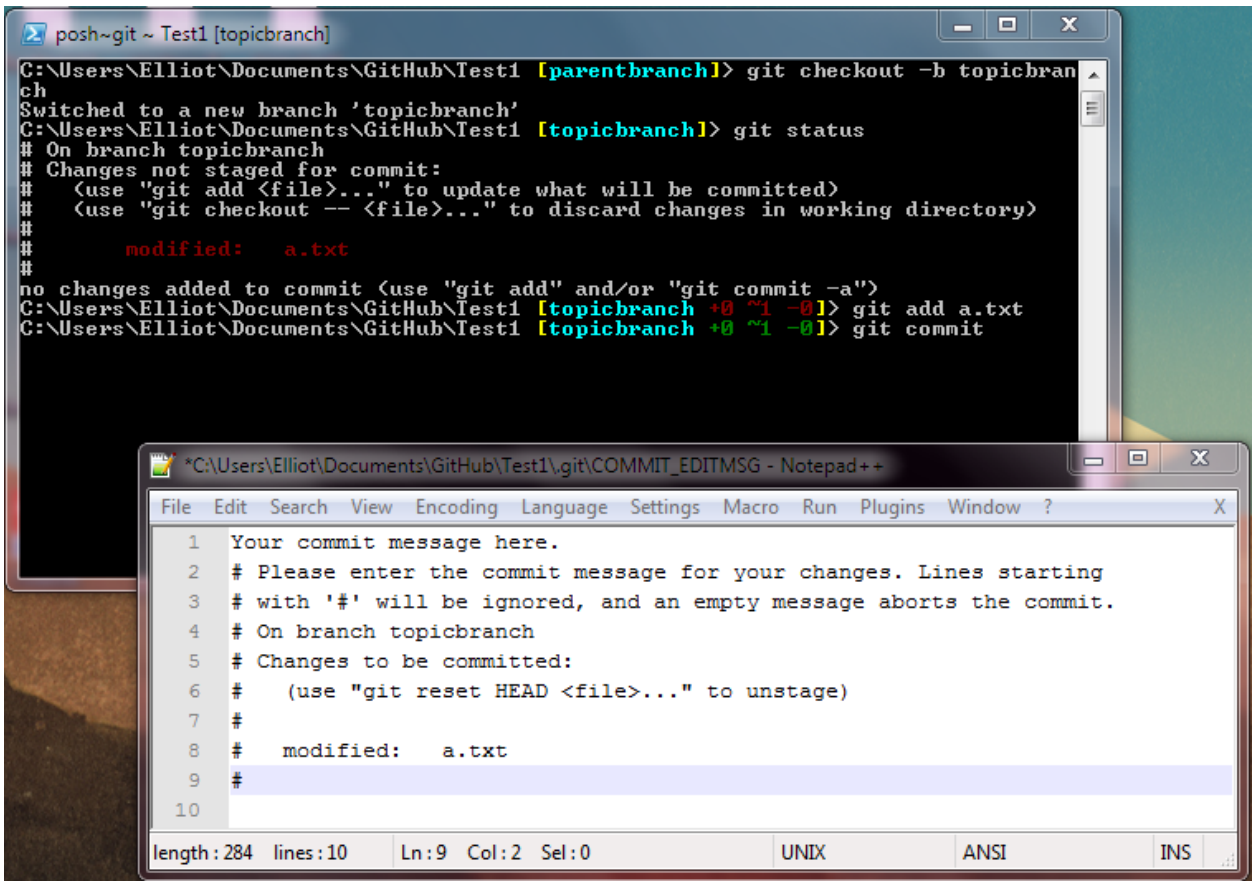
1. Checkout the parent branch and update it.
    a. `git checkout parentbranch`
    b. `git pull`
2. Create and checkout a local topic branch
    a. `git checkout –b topicbranch`

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.

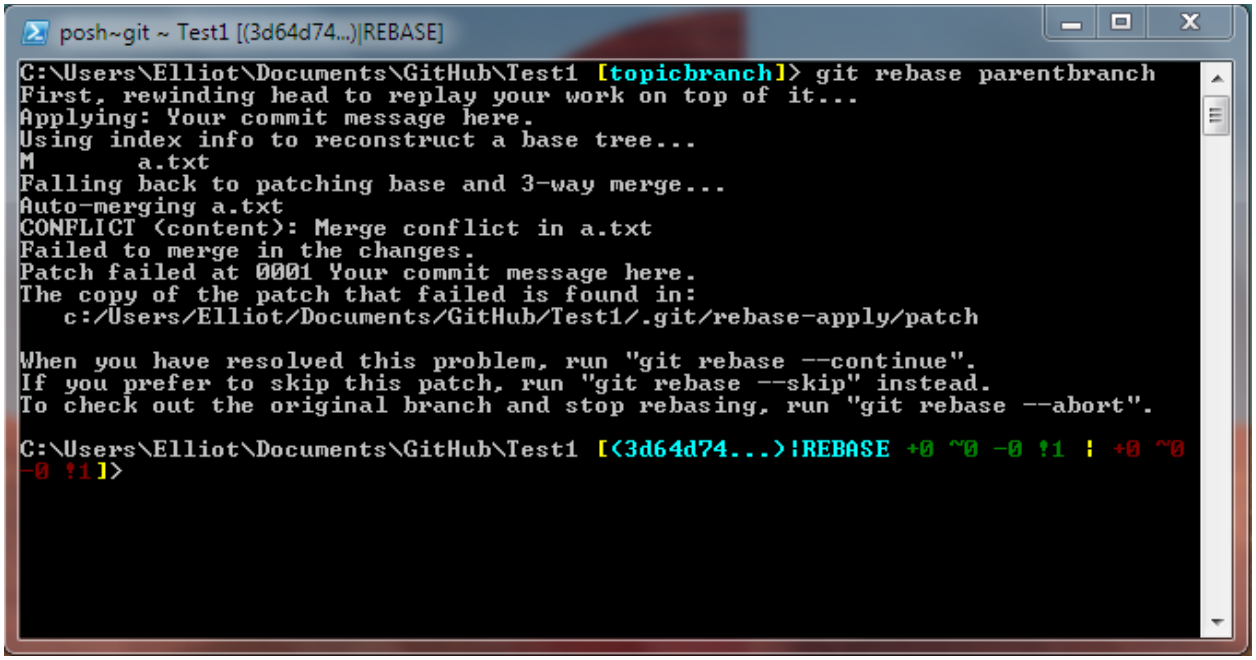
2. Stage changes you want to commit.
    a. `git add a.txt`
3. Create commit.
    a. `git commit`
    b. Git should automatically launch your text editor.
    c. Write a commit message
    d. Save the file and close the text editor.



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.*



Rebase conflicts—failed to auto-merge

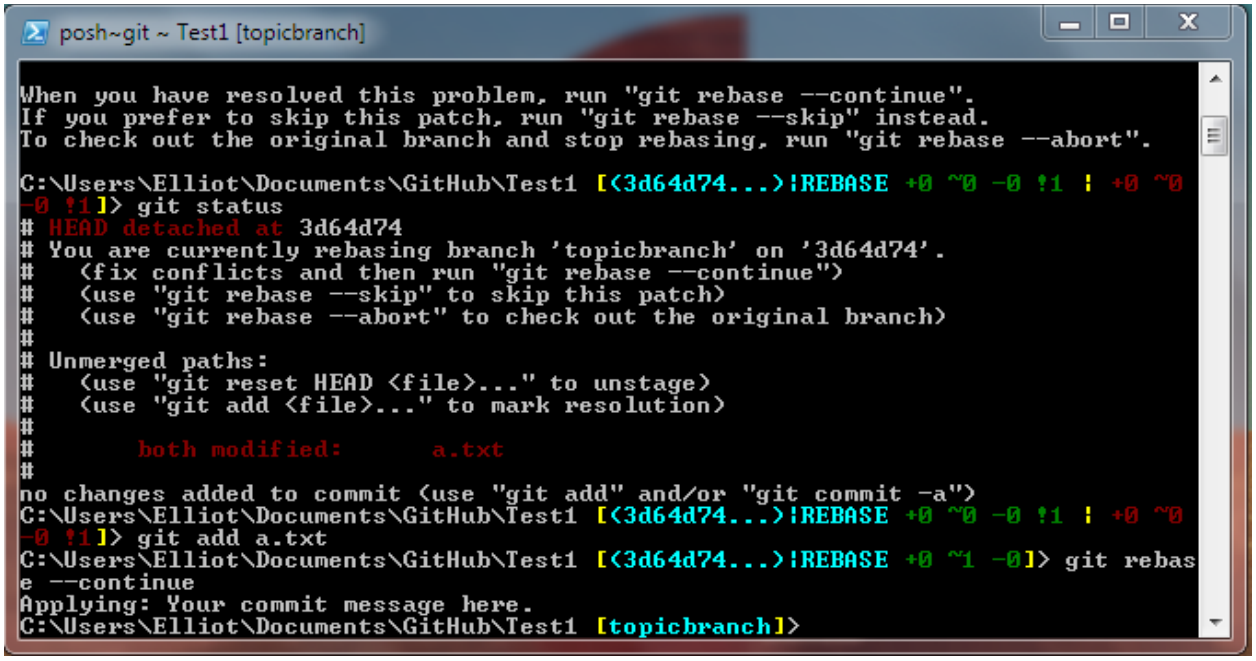
d. Resolve Rebase Conflicts

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`



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`