Branching and merging

1. To get started, review the slides on branching.

2. We will now create our first branch but before we do so, let us view how many branches are available in our repository. To do this, run the command `git branch`. This lists out all available branches.

3. To create a branch, decide what name you want to call your branch and then pass this name as an argument to the git branch command. For example, to create a branch called newbranch, run the command `git branch newbranch`. Git creates the branch but does not switch to it. Note that this new branch would be a replica of the branch from which the command was issued.

4. Now let's look again at the list of available branches. Run the command `git branch`.

5. We can now switch to this new branch by using the git checkout command as follows `git checkout newbranch`. All files for the current branch will be removed from the working directory and replaced with the files for the branch we have just swapped to. Note that you will not be able to switch to another branch if you have uncommitted changes so commit or discard all changes before switching branches.

6. To rename a branch, run the command `git branch -m <oldbranchname> <newbranchname>` using the old and new branch names.

7. To delete a branch, run the command `git branch -d <branchname>`. Git does some checks before deleting a branch.
    a. You can't delete a branch you are currently on.

b. If deleting the branch will result in a loss of data, it will issue a warning.
c. To go ahead, use the capital letter –D option to delete.

8. Merging involves updating a branch to have the changes of another branch. Note that the receiving branch must be checked out, that is, the merge command is issued from the receiving branch.

9. At this stage, you should have two (or more branches), that is, one branch called master, and one or more other branches. If you do not have two or more branches, create a new branch from the master branch as explained in step 3 (ensure you are already on the master branch before doing this).

10. Switch to this new branch (using the command in step 5).

11. Now make some changes to a file in this new branch, add the changes to the staging area and commit the changes.

12. We will now merge these changes to the master branch. To do this, switch to the master branch `git checkout master` and then run the command `git merge newbranch`
You should see a message similar to the one below.

```
Updating a3580bc..c21f858
Fast-forward
 Welcome.txt | 2 ++
 1 file changed, 2 insertions(+)
```

This type of merge is called a fast forward merge because the receiving branch is already a subset of the branch we are merging from.

13. If we made different changes to the master branch and the `newbranch`, Git may not be able to do a fast forward merge and sometimes, conflicts may arise, especially when different changes have been made to the same file in different branches. To see such conflict in action, let follow the next few steps.

14. Switch to the master branch and make a change in one of your files. Note the line number you have changed.

15. Add and commit that change to the repository.

16. Now switch to the newbranch, make a different change to the same line number of the same file and add and commit that change to the repository.

17. Now switch back to the master branch and run the command `git merge newbranch`.

18. A merge conflict should arise from this scenario.

```
Auto-merging Welcome.txt
CONFLICT (content): Merge conflict in
Welcome.txt
Automatic merge failed; fix conflicts and then
commit the result.
```

19. Now open that particular file using a suitable operating system program, like notepad, and locate the part or parts of that file that have multiple angled brackets, like this below

```
<<<<<<< HEAD
I am adding this from master.
=======
I am adding this from newbranch.
>>>>>>> newbranch
```

20. You should be able to see the changes made at both branches.

21. Now select the line you want to keep, remove every other text you do not need anymore (such as the angled brackets) and save the file.

22. Add the file to the staging area and then commit the change.

Using remote repositories (proceed only if you do not mind creating an account with GitHub)

23. First, we need to create an account with a version control hosting service such as GitHub. Visit GitHub.com and create an account (if you do not have an account with any version control hosting service).

24. Once you have created an account, you need to create a remote repository on github.

25. Once you have created the remote repository, you may choose to follow the steps on GitHub to add this repository to your local repository.

26. Substitute the words in angled brackets and and run the command `git remote add <remoteAlias> https://github.com/<username>/<repo>.git` to add the remote repository to Git.

27. To make your first push to the remote server, use the command `git push -u <remoteAlias> <branchName>`. This pushes the specified branch to the remote repository specified by the remote alias.

28. Any subsequent push will no longer require the command in step 27. We can simply run the command `git push`. Note that you cannot push changes to a remote repository that has been updated since the last pull. Changes made by other users in a remote branch must be brought down before new changes can be pushed by a user.

29. To pull down changes from the remote repository, run the command `git pull <remote branch>`

30. Repositories can also be cloned by using the command `git clone <repository url>`. See the slides on branching, merging and remote repositories for more information.