Case Study Git

- ## Local Operations :-

1] Create a new local git repository on to your machine and configure it for

user.name and user.email attribute.

2] Create a files in directory (File1, IgnoreFile and FileToDelete provided), say directory contains – File1.txt, IgnoreFile.txt and FileToDelete.txt, now start tracking only the File1.txt and FileToDelete.txt, and ignore the IgnoreFile.txt by using .gitignore.

3] Commit these changes to repository.

Use commit message as “This is First Commit, Tracking File1 and FileToDelete”

Note: - only File1.txt and FileToDelete.txt should be tracked and not the IgnoreFile.txt

4] Make 2 more commits by making following changes in the File1.txt

Add changes to file as:-

Change1 – commit with message as “Change1 in File1”

Change2 – commit with message as “Change2 in File1”

And if you check the log it should display in following manner:-

Change2 in File1

Change1 in File1

This is First Commit, Tracking File1 and FileToDelete

5] Delete the FileToDelete.txt file from directory and commit with message as “FileToDelete is deleted”

6] Display logs:-

- Show all the commit logs.
- Show last three logs in one line
- Show only commits that occur between from date and to date
- Show only those commit message that contains “File1” as in commits.

7] Permanently undo a “FileToDelete is deleted” commit snapshot.

8] Rename File1.txt to TestFile.txt

- ## Branching and Rebasing

1] Create a branch ‘bug’, checkout to bug branch to solve the bug in file1.

Bug is – Put a “ ”(space) between numerical and letters

e.g.:- Change1 → Change 1

2] Commit this change with message as “Bug Resolved – Space added between numerical and letters”

3] Merge the changes to master assuming bug is resolved.

4] Delete branch ‘bug’

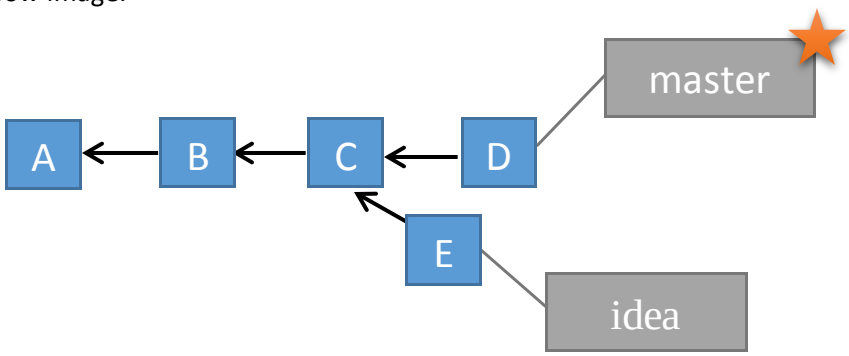
5] Recover the deleted branch ‘bug’ and rename it to ‘bug123’

6] Consider the following scenario and achieve the same to Rebase:-

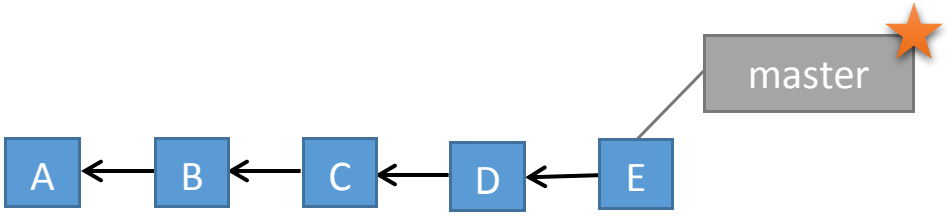
- Create new branch ‘idea’ and make a new commit on ‘idea’ branch

- Now checkout to master and make a new commit on master

Refer the below image:-



Now use Rebase to get the linear story line as follows:-



8] Now say bug is resolved and you are good to release this stable version, tag this release as ‘v1.0’ with message as “Stable version 1.0 released”.

9] Consider the following scenario to work with Stashing:-

Checkout to ‘bug123’ branch first and add text as “Change 3” in File1.txt, stage this change and don’t commit the same so that you can get a dirty state of working directory.

Now assume that you have a priority task to be made on master branch, checkout to master and store this dirty state (on ‘bug123’) to stash so that you get back to this state later.

10] Display the list of all stash.

11] Checkout to ‘bug123’ branch, reapply the stash which is stored recently and finally commit this change with the message as “Change 3 in File1”

• Git Remote:-

1] Create (Signup) GitHub account by visiting https://github.com/

2] Create new repository as “TestRepo”.

3] Add remote on to your local repository and name it as ‘Origin’.

4] Display all remote names with their details.

5] Rename remote branch ‘Origin’ to ‘RemoteBranch’.

6] Take a copy (Clone) of remote repository to your local machine.

7] Now create a new file as ‘RemoteFile.txt’ on ***remote*** repository, make 1 commit by making following change in the RemoteFile.txt

Add changes to file as:-

Change1 – commit with message as “Change1 in RemoteFile”

And if you check the log on ***remote*** it should display in following manner:-

Change1 in RemoteFile

Note: - If you check log on local there will be no commits which are made on remote.

8] Fetch remote ‘Origin’ to synch up with remote.

9] Finally merge the remote changes to local master.

10] Now, make 1 more commit on ***remote (GitHub)*** by making following change in the RemoteFile.txt

Add changes to file as:-

Change2 – commit with message as “Change2 in RemoteFile”

And if you check the log on ***remote*** it should display in following manner:-

Change2 in RemoteFile

Change1 in RemoteFile

11] Use Pull to take these recent changes to local.

12] Now, make 2 new commits on ***local*** by making following change in the RemoteFile.txt

Add changes to file as:-

Change3 – commit with message as “Change3 in RemoteFile”

Change4 – commit with message as “Change4 in RemoteFile”

And if you check the log on ***local*** it should display in following manner:-

Change4 in RemoteFile

Change3 in RemoteFile

Change2 in RemoteFile

Change1 in RemoteFile

13] Push these recent changes to remote.