Assignment 3

Introduction

The goal of this assignment is to consolidate skills in advanced Git commands and GitHub operations. It is assumed that you are up to date with concepts and commands taught in previous modules (Modules 1 to Module 4). This assignment focuses on the following topics:

- Git Branching and Merging, and techniques
- Commit History rewrite operations such as
    - Amendment of Git commits
    - Rebase
    - Pull with Rebase
    - Reflog
- Git Tags

Problem Statement

This assignment is a 9-step challenge and is the most detailed assignment of all.

- **Step-1**: Here you are required fork and clone a GitHub repo, whose link has been provided in the detailed steps

- **Step-2**: In this step, you are required to create a fresh new branch, display the list of all the local & remote branches in the repo and rename the branch

- **Step-3**: This step requires you to find a git diff between the feature branch (created in the previous step) and the master branch. Once diff (a git best practice) is done, you are required to perform a fast-forward merge of the two branches.

- **Step-4**: This step requires you to find a git diff between the feature branch and the master branch. Once diff is done, you are required to perform a no-fast-forward merge of the two branches.
- **Step-5**: This step requires you to find a git diff between the feature branch and the master branch once you have made changes in feature and master branch where both of the branches have diverged unlike the previous steps related to fast-forward and no-fast-forward merges. Once diff is done, you are required to perform a 3-way merge of the two branches.

- **Step-6**: Here you are required to use the "--no-edit" option to amend the HEAD i.e, the most most recent commit

- **Step-7**: Rebase helps to have clean and flat commit history. In this step, you are required to create a fresh new branch, make changes and commit those changes. Once commit is done, you are required to rebase your changes with master branch. Finally you need to examine commit history to confirm if the rebase went fine or not.

- **Step-8**: Reflog can immensely helpful to understand the transitioning of the HEAD through the history of a repo and to take corrective actions if needed. Here you need to do a time travel at a certain point of time in the commit history.

- **Step-9**: Finally you need to deal with Git Tags. In this assignment we have kept things simple and limited the exercise to lightweight tags only. You need to create a lightweight tag and display the list of tags in the repository.

Note: The yellow highlighted words which you will find in the steps below indicate the current git branch you are on.

Approach: Follow the steps and complete the assignment

1. **Step-1: Initial repository Setup**
    a. Fork the following public GitHub project - https://github.com/bibroy/insh-assignment-3
    b. Once you have forked project insh-assignment-3, simple clone the repository in your local repo
    c. Change (cd) to project directory insh-assignment-3

2. **Step-2: Git Branching**
    a. (master) Create a Git branch named branch1
    b. (master) Ok, it seems that's too trivial a name. Rename branch1 to feature-branch-1
    c. (master) Now display the list of all local branches
    d. (master) Now display the list of all branches

3. **Step-3: Fast Forward Merge**
    a. (master) Checkout the branch feature-branch-1

b. (feature-branch-1) Edit robots.txt (you can use vi editor or any text editor that you have configured as your default text editor). Make a simple change in the file. Let's call it simple-change-5
c. (feature-branch-1) commit the simple-change-5 (let's say the commit message is "commit message 5"; your commit message can be anything whatever you want it to be)
d. (feature-branch-1) checkout master branch
e. (master) Do a git diff between feature-branch-1 and master branch since we will merge the changes made in feature branch (best practice)
f. (master) Now do a ***fast-forward merge*** of feature-branch-1 with master branch. Confirm that the merge went through successfully from the message git displays.
g. (master) Run git log command

4. **Step-4: No Fast Forward Merge**
   a. (master) Checkout the branch feature-branch-1
   b. (feature-branch-1) Edit robots.txt (you can use vi editor or any text editor that you have configured as your default text editor). Make a simple change in the file. Let's call it simple-change-6
   c. (feature-branch-1) commit the simple-change-6 (let's say the commit message is "commit message 6"; your commit message can be anything whatever you want it to be)
   d. (feature-branch-1) checkout master branch
   e. (master) Do a git diff between feature-branch-1 and master branch since we will merge the changes made in the feature branch (best practice)
   f. (master) Now do a ***no-fast-forward merge*** of feature-branch-1 with the master branch. Confirm that the merge went through successfully from the message git displays. If all went successfully, you will notice a new merge commit.
   g. (master) Run git log command

5. **Step-5: Three-way Merge**
   a. (master) Checkout the branch feature-branch-1
   b. (feature-branch-1) Edit robots.txt (you can use vi editor or any text editor that you have configured as your default text editor). Make a simple change in the file. Let's call it simple-change-7
   c. (feature-branch-1) commit the simple-change-7 (let's say the commit message is "commit message 7"; your commit message can be anything whatever you want it to be)
   d. (feature-branch-1) checkout master branch
   e. (master) Edit humans.txt (you can use vi editor or any text editor that you have configured as your default text editor). Make a simple change in the file. Let's call it simple-change-8

f. (master) commit the simple-change-8 (let's say the commit message is "commit message 8"; your commit message can be anything whatever you want it to be)
g. (master) Do a git diff between feature-branch-1 and master branch since we will merge the changes made in the feature branch (best practice)
h. (master) Now do a ***3-way merge*** of feature-branch-1 with the master branch. Let the merge commit message be "3-way merge commit message". Confirm that the merge went through successfully from the message git displays. If all went successfully, you will notice a new merge commit.
i. (master) Run git log command.
j. (master) We are done with feature-branch-1. Now delete the feature branch
k. (master) Run git log command

6. **Step-6: Amending commits**
    a. (master) Edit robots.txt (you can use vi editor or any text editor that you have configured as your default text editor). Make a simple change in the file. Let's call it simple-change-9
    b. (master) Add your changes to staging
    c. ((master) commit the simple-change-9 (let's say the commit message is "commit message 9"; your commit message can be anything whatever you want it to be) but this time run this command so that we can amend the most recent commit. Here amending means, the most recent commit should be amended in such a way that replaced the most recent commit with a brand new commit but does not result in changing the commit message

7. **Step-7: Rebasing a branch**
    a. Create and checkout to a new branch called feature-branch-2
    b. (master) Checkout the branch feature-branch-1
    c. (feature-branch-2) Edit robots.txt (you can use vi editor or any text editor that you have configured as your default text editor). Make a simple change in the file. Let's call it simple-change-10
    d. (feature-branch-2) commit the simple-change-10 (let's say the commit message is "commit message 10"; your commit message can be anything whatever you want it to be)
    e. (feature-branch-2) checkout master branch
    f. (master) Edit humans.txt (you can use vi editor or any text editor that you have configured as your default text editor). Make a simple change in the file. Let's call it simple-change-11
    g. (master) commit the simple-change-11 (let's say the commit message is "commit message 11"; your commit message can be anything whatever you want it to be)
    h. (master) checkout feature-branch-2

i. (feature-branch-2) Run command to rebase feature-branch-2 based on the master
j. (feature-branch-2) checkout to the master branch
k. (master) Run git log command
l. Run git command to merge feature-branch-2
m. Run the git log command again to see the effect of rebasing

8. **Step-8: Reflog**
    a. Run appropriate command to display all the entries in Reflog (hint: type “q” to quit if the reflog entries are too many to accommodate in single screen)
    b. Run command to see the fifth prior value of the HEAD of your repository
    c. Display reflog information for master branch formatted like the git log output

9. **Step-9: Tags**
    a. Create a lightweight tag named "v1.4-lwtag"
    b. List out available tags in the repo
    c. Display tag details for the tag named "v1.4-lwtag"

Submission

After completing the assignment, upload the text solution file. This file should contain the following:

- All git commands run for each of the steps of the assignment/project. Each command should start with “$” + single space
- Each command should be preceded by a line (or lines) of comment starting with a “#” + single space. The comment should describe the purpose of execution of the command.
- If the assignment/project involves working with GitHub, describe the steps taken in a step by step manner. The steps can be explained using comments (beginning with a “#” + single space)
- Each step mentioned in the assignment should be separated from other steps by few blank i.e new lines. You need to mention the step number, optionally with a description mentioning the purpose..

Solution

You can download the solution to the assignment from the progress report once you have uploaded the solution. You may compare your solution with the provided one.