GIT-BASICS

Please make sure to read the whole document before getting started and to follow the instructions we provide to the letter (e.g., use the exact commit messages provided in the assignment, rather than variations of them). If you don't know how to accomplish a task, either consult Git's help by running "git --help <command>" or leverage online resources (there are plenty, such as this Git cheat sheet). If you receive an error while executing a Git command, make sure to read the error message–Git often suggests exactly the right thing to do.

REPO :
your personal github repo

If you are not familiar with Git and feel completely lost, we suggest to take the tutorial at http://try.github.io/levels/1/challenges/1, which should further get you familiar with Git's basic concepts.

We also suggest that you use GitHub's "Network" view to monitor the state of your local and remote repository throughout the assignment. To do so, go to the GitHub page for REPO, select menu "Graphs" on the right, and then select the "Network" tab at the top. In this way, you will have a visual representation of how the repository evolves, which can be very useful for better understanding Git and how it operates. In addition, we provide a screenshot of how your repository should look at the end of the assignment. Alternatively, you can run "git log --graph" from the command line, which outputs a text-based representation of the same information.

If you were to make a mistake while working on REPO, however, and you wanted to restart from a clean slate, in most cases you should be able to do so by executing the following instructions:

- Run "git log" from within your local repository
- Get the last commit ID in the list (i.e., the one with the earliest date, which should have "Initial commit." as its associated comment.)

- Run in each of the two terminals, and from the root of your repo
    - git checkout master
    - git branch -D development temp
    - git push origin :development
    - git push origin :temp
    - git tag -d V1
    - git push origin :V1
    - git reset --hard <last commit ID>
    - git push --force

- IMPORTANT:
    - These are instructions that you should follow only if you mess up REPO.
    - This is one of the few cases in which we recommend the use of the “git push --force” command, which we otherwise strongly discourage, as it can have disastrous effects.
    - Some of the above commands may fail if you haven't yet done what they are trying to undo. These errors can be safely ignored.

- After resetting the state, there is no need to re-clone the repository (i.e., there is no need to perform the first 5 steps of the instructions).

Instructions:

Part 1 (Terminal 1)

Before you start, make sure to specify your name and email address using command “git config”, if you haven’t already.

1. Open a terminal window

2. Create and go to directory User1

3. Clone REPO

4. This should create a directory called **“devops”** under directory User1.

5. Go to directory **“devops”** (here you can also open the Network view in GT GitHub and start monitoring how your repository evolves)

6. Create and go to directory **GitBasics**

8. Create a file called myinfo.txt that contains only one line with your first and last name

9. Commit the file to your local repo with comment “Added myinfo file”

10. Create a branch called “development” and switch to it

11. Create a file called dev1.txt that contains the text “Dev 1 file”.

12. Commit the file to your local repo (it should be in branch “development”) with comment “Added dev1 file”

13. Switch to the “master” branch

14. Edit file myinfo.txt and add your zip code in a separate line (feel free to make up the code, if you don't want to use yours)

15. Commit the file to your local repo with comment “Edited myinfo file”

16. Merge the “development” branch into the “master” branch with commit message “Merge #1”

17. Push all branches to the remote repository

Part 2 (Terminal 2)

1. Open a second terminal window

2. Create and go to directory User2

3. Clone REPO

4. Just like before, this should create a directory called **devops** under directory User2

5. Go to directory devops/GitBasic

6. Switch to the “development” branch

7. Create a file called dev2.txt that contains the text “Dev 2 file”.

8. Commit the file to your local repo (it should be in branch “development”) with comment “Added dev2 file”

9. Create a branch called “temp” and switch to it

10. Create a file called mytemp.txt that contains the text “Mytemp file”.

11. Commit the file to your local repo (it should be in branch “temp”) with comment “Added mytemp file”

12. Create and commit to branch “development”, with the usual comment, a file called dev3.txt that contains the text “Dev 3 file”.

13. Merge the “temp” branch into the “development” branch with commit message “Merge #2”

14. Merge the “development” branch into the “master” branch with commit message “Merge #3”

15. In the master branch, edit file myinfo.txt and add your favorite soccer club
16. Commit the file to your local repo with comment “Edited myinfo file again”

17. Push all branches to the remote repository

Part 3 (Terminal 1)

1. Go back to the first terminal

2. Make sure to be in branch “master”

3. Edit file myinfo.txt and add your phone number in a separate line (use a fake phone number)

4. Commit the file to your local repo with comment “Edited myinfo file for the third time”

5. Push the master to the remote repository. To do so, you will have to suitably pull changes from the remote repository and handle conflicts. In handling conflicts, make sure not to lose any content, not to have any of the extra text added by Git to mark the conflicting

parts, and to preserve the order of the information as it appears in the doc (i.e., first and last name, zip code, fav soccer club, and phone number). Use commit message “Final merge with conflicts fixed” for the additional commit after merging and handling the conflicts.

6. Tag the current version of the master as “V1” and push the tag to the remote repository. Use a lightweight tag.

At this point you are done, and your remote repository should look like the one in the figure below in the "Network" view on GitHub. If it doesn’t, it means that you made a mistake in one of the steps. We expect the network view to match the figure below exactly and all the commit messages and the content of the files in the repo to be correct.