SIMON FRASER UNIVERSITY
Department of Computing Science
CMPT 276: Assignments

Assignment 1: Working with Git

1 Deliverables

This is an individual assignment. You will be working on your group repository; however, each team member has to complete the assignment separately.

The TA will mark the history and outcome of the assignment on GitLab on the specified deadline. See the Schedule page on Canvas for deadlines.

Follow the steps below to make your environment ready for the lab work.

2 Setting up git

2.1 Git/GitLab

Study the following resources for learning git if necessary:

- Git Reference: `https://git-scm.com/book/en/v2/Git-Internals-Git-References`
- Git tutorial: `http://gitimmersion.com`
- Git cheat sheet: `https://www.atlassian.com/git/tutorials/atlassian-git-cheatsheet`
- Handling merge conflicts: `http://www.gitguys.com/topics/merging-with-a-gui/`

There are multiple ways you can work with Git.

1. Use the terminal. For example: `git status -s`
2. Use EGit in Eclipse or IDE of your choice. For instance, see `http://eclipse.github.com` for instructions for Eclipse.
3. Use a Git client: `https://git-scm.com/downloads/guis` or `http://www.sourcetreeapp.com`.

2.1.1 Most frequently used Git operations:

- `$ git clone REPOS`: clone the remote repos.
- `$ git pull origin master`: get all the changes from remote repos.
- `$ git status -s`: what has changed?
- `$ git add FILENAME` or `git add .`: add changes in my working directory to my local Git repos.
- `$ git commit -m "my changes include..."`: commit all the added changes to my local Git repos.
- `$ git push origin master"`: push all my committed changes to remote repos.

2.2 Basic Git Operations

Each group has a dedicated private repository on GitLab. *These repositories belong to SFU/CS and are not to be made public or shared with anyone outside of your group.*

1. Clone your group's repository onto your machine, e.g.:

   `$ git clone git@csil-git1.cs.surrey.sfu.ca:276-group-<group#>/project.git`

   (You need to create an SSH to access your repository through SSH: `https://docs.gitlab.com/ee/ssh/`.)

2. Open (create) the README.md file. Add your name, email address, and a sentence describing what you want your teammates to know about you.

3. Stage the README file (e.g., `$ git add README.md`).

4. Commit the changes to your local repository with an appropriate commit message, explaining what you are committing (`$ git commit m 'Your message goes here. Write a descriptive one.'`).

5. Push the changes to the remote repository (`$ git push origin master`).

   Note: You need to pull the latest changes to the repository first if your teammates have pushed new updates in the meanwhile (`$ git pull`).

2.3 Working with Branches

1. Create a new branch called <yourFirstName-dev> (if two+ people in the group have the same name, add an identifier of your choice to the end of your name).

   `$ git branch <branchName>`

2. List all of the branches in your repository: `$ git branch`.

3. Switch to your new branch: `$ git checkout <branchName>`.

   Note: You can simultaneously create and check out <branchName>: `$ git checkout -b <branchName>`.

4. Add and commit a new file named test1-<yourName>.txt to your new branch.

5. Switch back to the `master` branch.

6. Add and commit a new file named test2-<yourName>.txt to the master branch.

7. Prepare to merge the branches.

   - The HEAD should point to the branch that is receiving the merge, in the case, the `master`. You can confirm this by checking out the branch (`$ git checkout master`).
   - Fetch latest remote changes. While you have been working on your repository, your teammates may have pushed new updated to the remote repository. Make sure that your branches are up to date by executing `$ git fetch`. Once the fetch is completed ensure the master branch has the latest updates by executing `$ git pull`.
   - Now you can merge your new branch into `master` by executing `$ git merge <branchName>`, which will merge <branchName> into the `master` branch, i.e., the receiving branch.
   - Push your changes to the remote repository. Check the state of the remote repository and the history of your commits on GitLab's web application. Can you see both new files? Can you see your new branch?
   - Long running branches:
     - Checkout your new branch. Push the changes in this branch to the remote repository: `$ git push origin <branchName>`.
       Check your repository on GitLab's web application. Can you see the branch now?

- For long-running branches, you will need to submit merge requests (aka pull requests) throughout the life cycle of your project. Read more on the topic here:
  `https://docs.gitlab.com/ee/user/project/merge_requests/getting_started.html`
  `https://docs.gitlab.com/ee/user/project/merge_requests/creating_merge_requests.html`

See `https://git-scm.com/book/en/v2/Git-Branching-Basic-Branching-and-Merging` for more information on branching and merging.