Git Practicals 2 - Using Git and Github - a comprehensive step-by-step tutorial of a typical workflow

1. If not already done, first Obtain and install the git tool from https://git-scm.com/downloads (As an option, you may also download and setup the Github desktop client app from - https://desktop.github.com/ or some other git GUI client app (e.g. SourceTree, GitKrakken etc.), if you like.

2. A couple of required global configs to do after installing Git:

    2.1 c:\> **git config --global user.name "YourName"**
    2.2 c:\> **git config --global user.email "your-email@some-domain.com"**

To Use Git with Github for collaborating on a repo:

1. Project leader will Login to github, (create a new account, if needed) and create/init a new repository for your CS425-SWE project.
2. Users can then Login to github (using their own Github account), visit the project's repository page and Create a Fork of the repository. This makes a copy of the repository for your own use, inside your github account.
3. On your local computer, make a new folder/directory for the project (if you do not already have one).
4. Open a cmd window/terminal and cd into the project folder/directory.
5. Clone your copy of the repository (i.e. the fork you've created in your github) on to the new project folder you've just created on your local machine, by doing the following:
    a. In your github account, with the forked repository open, click on the green button named, "Clone or download", and click on the 'Copy to clipboard' icon.
    b. On the cmd line window/terminal on your local machine, run the command, $ git clone [followed by the url to the repository, pasted here] e.g. c:\project-folder>*git clone https://github.com/okalu/cs425-project1.git*. This clones the repository, and produces a new folder for it containing all the source files downloaded from your github.
6. Next, Create a new branch by doing the following:
    a. Cd into the new repository folder/directory on your local machine.
    b. Create a new branch of the project, using the git checkout command, as follows: $ git checkout -b <name-of-the-branch>. E.g. c:\project-folder\repo-folder>git checkout -b add-some-new-feature or c:\project-folder\repo-folder>git checkout -b fix-some-existing-feature. Note: the name of your branch serves to convey the purpose or task which your branch is intending to accomplish. Just a way to identify our branch. Note2: Every repository has a main/default branch which is named, master. Running the git checkout command, simply switches from the

master branch to your new branch name. So therefore, any new additions or changes you make will be made on the branch and not the master.

   c. With the branch created, now you can begin working on your branch of the project/repository; editing existing source files, adding new files, packages or libraries etc.

7. Make any necessary changes and add and commit those changes (for example, add a new source file, open and edit any existing source file etc.) to the branch of the repo, by doing the following:
   a. On the cmd line terminal, execute the command, $ git status, to show/see the files that have been added or changed.
   b. Add all the changes to the branch by executing the command, $ git add . (note: this adds all changed files. You can also run, $ git add <filename.ext>, to add only a specific file or use wildcards to pick and choose which files to add).
   c. Now, commit the changes by running, $ git commit -m “<a short description of the commit” e.g. c:/project-folder\repo>git commit -m “Added some new feature in index.html”
8. Push your changes to Github: (note: commit only updates the repo database on your local machine; it still needs to be pushed up to the repo on your github account.) To do this, run the git push command as follows - $ git push origin <branch-name> e.g. c:\project-folder\repo-folder>git push origin some-new-feature
9. **Submit your changes for review**: Having implemented your changes to the project, it is now time to request for your changes to be reviewed and perhaps be incorporated into the original source repository and hence the software product. To do this, you create and send a Pull Request. For this, do the following:
   a. Go over to the repository on your github account, click on the button named, "New pull request". Fill-in the ‘Open a Pull Request’ form and submit it, to create a pull request.
   b. The maintainer(s) of the master branch of the original source repository, will review your changes and merge them into the master branch. And you’ll receive an email notification when this is done.
10. Synchronizing your fork with the original source repo: (note that the master branch of your fork will not have the changes you have made and had merged by the maintainer(s)). To update and keep your fork in-synch with the original source repo and update your fork’s master branch, do the following:
    a. Switch back to the master branch by executing the command - $ git checkout master
    b. Add the original source repository’s url as upstream remote url, by running the cmd, **$ *git remote add upstream https://github.com/okalu-cs425/cs425-project1***. This tells git that another version of this project exists at the specified url and we’re calling it ‘upstream’.
    c. Once the changes submitted via the pull request (PR) have been merged, you can then fetch the updated version of the original source repo (named,

upstream), by executing the command, ***$ git fetch upstream***. This fetches all the changes that exist in the upstream remote.

d. Now, to merge the new/latest revision of the original source repo (i.e. upstream/master) into your local master branch, execute the following cmd, ***$ git rebase upstream/master***. This applies all the changes earlier fetched, to the master on your local machine.
e. To update the fork on your github account, you simply push your updated master branch from your local machine to the fork on your github, by executing the cmd, **$ git push origin master**. Note that this is effectively saying, to do a push to a remote named, origin, and to the branch named, master.

11. At this point, changes you made to the branch on your local machine copy, have been merged to the master branch of the original source repo (i.e. upstream/remote) and you have also merged the master branch of the original source repo into your own master branch (both on your local machine and on the fork in your github account). So therefore, the branch, both on your local machine and on github, is no longer needed, since it is now just a duplicate of what you already have in the master branch at both places.
12. Delete the branch: You may then delete the branch by running the cmd, $ git branch -d <branch-name>. You can also delete the version of it in the remote repository too, by executing, $ git push origin --delete <branch-name>.
13. That's it!!! Use of Git-and-Github, in a full round-trip, from repo cloning to sending a PR to synching with upstream/remote.
14. **Note**: Any of the following other alternative dev tools/IDEs can also be used to accomplish all of the above - Github Desktop app, Visual Studio 201x or Visual Studio Code, GitKrakken, SourceTree by Atlassian, Eclipse, Netbeans, IntelliJ IDEA, STS etc.
15. **//-- The End --//**