COEN 174 Software Engineering Fall 2021

Lab 3 (60 pts)

In this lab, you will learn to use Git and GitHub to manage versions of your files.

Hopefully, all of you have completed PreLab3 which requires you to install Git on your laptop and create an account on GitHub.

You will do three exercises in this lab, using the directions and references given below.

What is a Version control system (VCS)?

* A VCS is a software tool that helps you manage changes to files over time.
* A VCS detects and records every modification made to the file in a special database.
* The changes can be recalled as versions of the file.
* Developers, especially working in teams, should place their source code and other related artefacts in a VCS.

Introduction to Git

What is GitHub?

* GitHub is a website and cloud-based hosting service that helps developers store and manage their code, as well as track and control changes to their code.

Some Useful Terms:

* Repository (repo): A location where the files and their histories are kept.
* Master Repo: Shared version stored on a server or GitHub
* Local Repo: your local copy that you can play with and make changes to. Only seen by you until you push your changes
* Checking Out: Obtain a copy of the files in the repository.
* Working Copy: The files that are checked out.
* Checking In (Commit): Return changed files to a repository.
* Branching: Files start from the same original copy and change in parallel.
* Merging: Combine branches into one common descendent.
* Clone: Copying an entire repo
* Push & Pull: Synch repos between local and remote copies.
* Conflict: When merging cannot be done automatically.
* Resolve: Fix the conflicts

Reference: <https://git-scm.com/book/en/v2/Git-Basics-Getting-a-Git-Repository>

**Exercise 1**

In this exercise, you will create a repository (repo) on your local machine and manage the files using Git.

Setting up the local repository (repo)

You can set up a Git repo in 2 ways.

* Take a local directory that is currently not under version control, and turn it into a Git repository, or
* You can ***clone*** an existing Git repository from elsewhere.

**At first, we will do the step in a).**

Create a folder called **lab2git** under **coen174labs, folder**.

cd to **lab2git** folder.

At the prompt, type git init

Then type, git status

What does it say on the console?

**On branch master**

**Initial commit**

**nothing to commit (create/copy files and use "git add" to track)**

Now, add a file to this folder. This is a text file with the title, **aboutMe.txt**. Include a brief description of yourself, for example, your name, major, year and your hobbies etc.

Type, git add aboutMe.txt.

Type git status

What does it say on the console?

**On branch master**

**Initial commit**

**Changes to be committed:**

**new file: aboutMe.txt**

Now, type git commit

What does it say on the console?

**It asked for a commit message and then:**

**[master (root-commit) 8813c3a] Commit 1**

**Committer: stephen a tambussi <stambussi@linux11101.dc.engr.scu.edu>**

**1 file changed, 1 insertion(+)**

**create mode 100644 aboutMe.txt**

The second file you will now add to the folder is **employees.txt** that you used in your Lab1. Copy the file to the folder.

Now type git status

What does it say on the console?

**On branch master**

**Untracked files:**

**empnames.txt**

Now, type git add employees.txt

Type git commit

Add the following change to the file, aboutme.txt. To the file, add a list of courses you are taking this quarter.

Type git status

What does it say on the console?

**On branch master**

**Changes not staged for commit:**

**modified: aboutMe.txt**

**no changes added to commit**

Now, type git add employees.txt

Type git commit

What does it say on the console?

**It won't let me commit and repeats the above status message**

**Note**: After every change, do git add before git commit.

Add a third file (any file of your choice) to the folder. Assume you don’t want this file to be tracked by git. What do you do?

**You do nothing. You do not do git add <filename> so that it is not tracked. Or, you can have a .gitignore file that you add the file of your choice to and git will not track it.**

**Exercise 2**

In this exercise, you will create a repository on GitHub using your account.

**Creating an empty folder (repo) on GitHub**

* Login to your Github account.
* At the top right of any Github page, you should see a '+' icon. Click that, then select 'New Repository'.



* Give your repository on GitHub, the same name as your local repo (**lab2git**) that you have created in exercise 1.
* Click 'Create Repository'.
* The screen you should be seeing now on Github is titled **'Quick setup — if you’ve done this kind of thing before'**.Copy the link in the input right beneath the title, it should look something like this: <https://github.com/.../lab2git> This is the web address that your local folder will use to push its contents to the remote folder on Github.
* Go back to your folder in the terminal/command line on the local machine.
* In your terminal/command line,

type git remote add origin <https://github.com/.../lab2git>

* Push your branch to Github: git push origin master
* Go back to the folder/repository screen on Github that you just left and refresh it. The title **'Quick setup — if you’ve done this kind of thing before'** should disappear, and you should see the files from your local folder there.
* Now, add a new file (of your choosing) to the repo on the GitHub.



* In your terminal/command line,

type git pull origin master

* Check the files in your lab2git folder on your local machine. Do you see the new file you have added on the GitHub repo?

**Exercise 3**

**Tanner Kaczmarek**

**Stephen Tambussi**

In this exercise, you will learn to collaborate using Git and GitHub.

When you work in teams on your projects, you should be able to work on separate copies of the code, make changes and the changes to the common repository.

For this exercise, two people will collaborate with one being the owner of the repo and the other collaborator and they switch roles.

**Owner**

* Owner invites the collaborator and gives access to the repo
* Go to GitHub repo (lab2git).
* Click on Settings tab.
* Click Collaborators.
* Enter your collaborator’s username (GitHub username)

**Collaborator**

* Create a folder called **lab2From** in a location of your choice, on your local machine.
* Go to <https://github.com/notifications> and accept access to Owner’s repo (lab2git)
* Using the terminal, clone the owner’s repo using the command,

git clone URL\_Of\_OriginRepoOfOwner *pathTol****ab2From***

Now, as a collaborator, you have a copy of **lab2git repo** from the owner in the **lab2From** folder.

* Go to **lab2git folder, open employees.txt file and add your name in the format below:**

*YourLastName*, *FirstName*:*a 5 digit number*:*a date*:Office *officeNo*

Save the file.

Now after you have completed the modification to the file, you must commit the changes to the local repo and push the changes to the owner’s repo on GitHub.

Commit changes to your local repo

git add employees.txt

git commit -m “changes from *yourname*”

Push your changes to the owner’s repo on GitHub

git push origin master

**Owner**

Owner now downloads, reviews and accepts the changes from collaborator.

Download collaborator’s changes to the Owner’s local repo

git pull origin master

References

<https://git-scm.com/book/en/v2/Git-Basics-Getting-a-Git-Repository>

<https://swcarpentry.github.io/git-novice/>

**Collaborating**: <https://swcarpentry.github.io/git-novice/08-collab/>