ECEN 350 GITHUB FOR LAB

Step 0 - Find the TA information for your section.

Name	Net ID	Lab Sections	Channel for Slack
Myung Seok	mrshim1101	501 (Honor), 503, 506	#section501_503_506
Sujoy	sujoy_saha	502, 504, 505	#section502 #section504 #section505
Bobae	bzk0029	507, 508, 510	#section507_508_510
Eric	ericgarfinkle	509, 511	#section_509_511

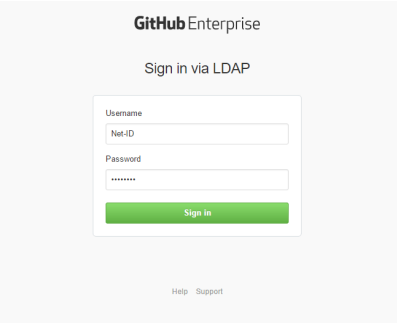
Step 1 - Get familiar with how GIT works.

Watch Video Lecture/Tutorial by following the link
https://www.youtube.com/watch?v=mYjZtU1-u9Y&list=PL1F56EA413018EEE1
This is just to give you an overview of the system - do not worry if you miss some details.

Step 2- Create online account on TAMU Github

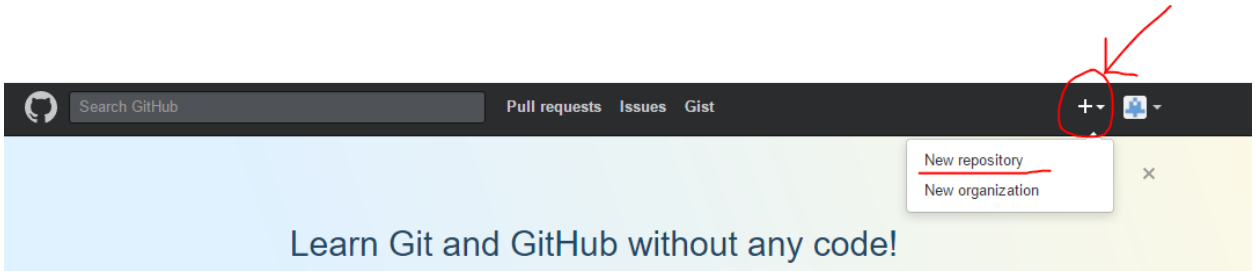
Steps for git setup(online):

1. Create a Github account:
    a. Go to https://github.tamu.edu/
    b. Enter Net-ID and Password

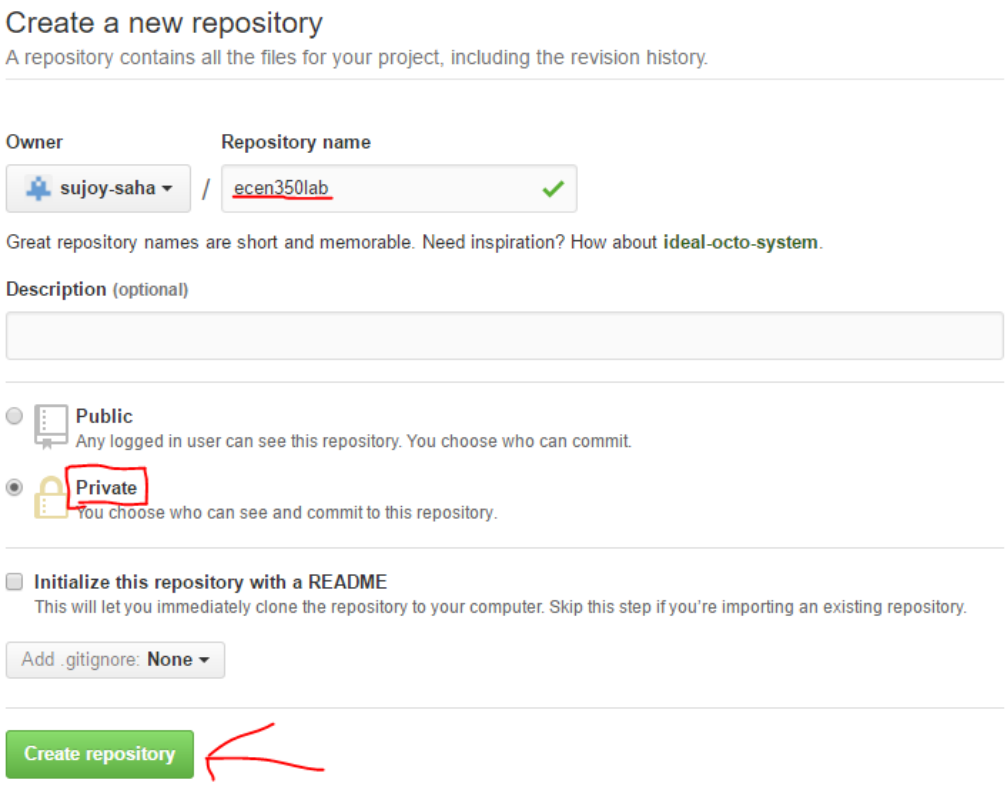


Step 3 - Create a new repo

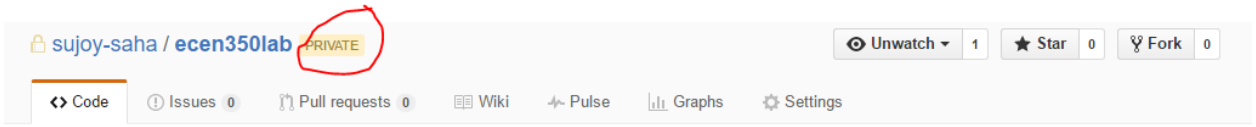
Now once you created the account, **create a new repo** (which is where you will store your code for the labs.



- Now create a new **[PRIVATE]** repo called "**ecen350lab**" as shown below. Your repo has to be **private** and have the name "**ecen350lab**".



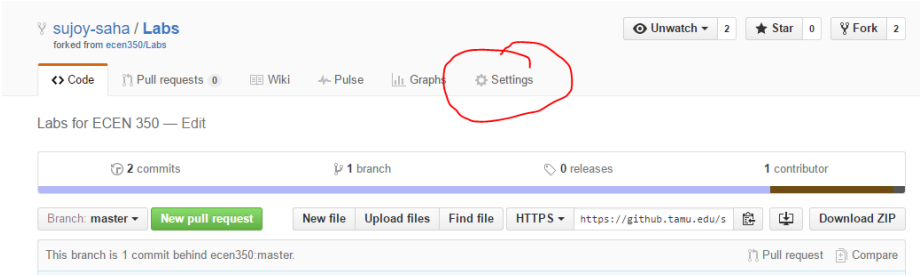
- Once you do this, you will get the following prompt:
  - Note: incase your NEIT-ID has a special character it is possible it will be replaced by a "-", in that case please do not change your username.



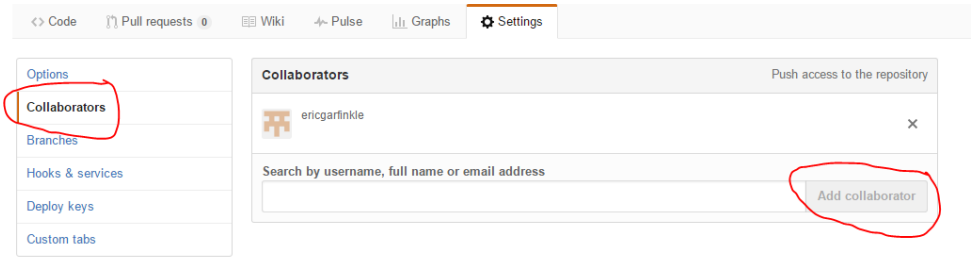
- Please create a new repo using the above method. Now take a look at **settings** in the menu bar.

- Now add your section TA as collaborators.

    - Step 1: Click on settings as shown below in the Labs repo:



    - Step 2: Here you will add your TA as a collaborator by using the NetID (check the list on ecampus or see the table in the first page of this document)

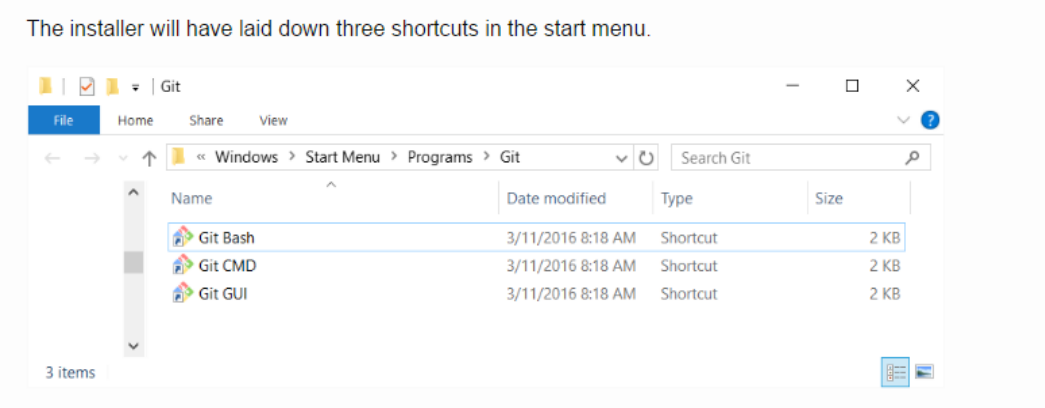


    - That's it for online setup!

Step 4 - Install Git for local copy (on your personal machine) **(optional)**

Git is already installed on class computers. Please complete this section if you would like to install it on your home machine. This step is optional.

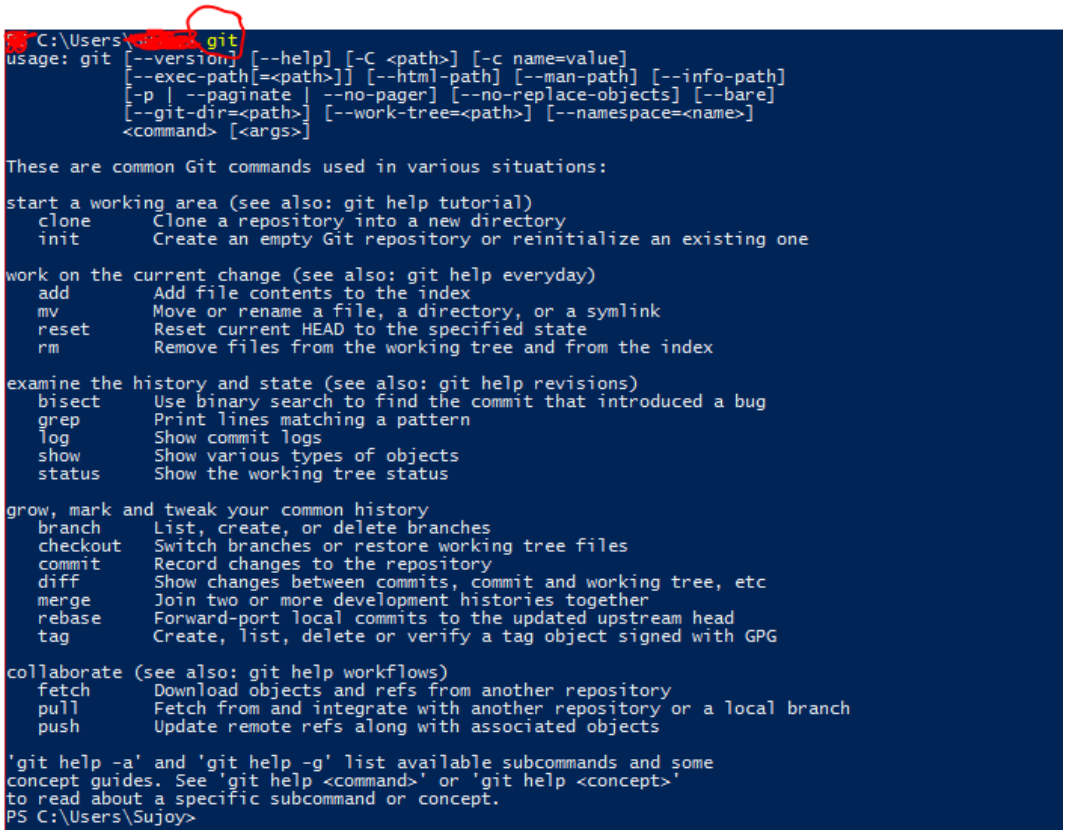
- For windows 10 follow: https://www.develves.net/blogs/asd/articles/using-git-with-powershell-on-windows-10/
    - [NOTE: Only follow the steps here till installing Git Bash.]
    - You will get the following:



    - **Note Git Bash, Powershell etc at this point should have Git. Please use Git Bash in case powershell does not work. All of these command line interfaces are the same**.

- For any other OS follow:
    - https://git-scm.com/book/en/v2/Getting-Started-Installing-Git

NOTE: in **Lab git should already be there**. Its Linux, so it will be available on the **command line**. Test: type ***git*** to check for git installation. You should get something similar to:



4. Local Git configuration settings**(in command line in linux/powershell/git bash in windows)**

- Step 1: Add username(net-id) using the following:
  - `git config --global user.name "NET-ID"`
- Step 2: Add email (tamu email only) using the following:
  - `git config --global user.email "your_email@tamu.edu"`

Step 5 - prepare your repo

If you skipped Step 4, this needs to be done in class (on the first lab)

- Clone your repo: Download the Labs repo into your local system. Do this via command line or powershell:
    - **git clone --bare https://github.tamu.edu/ecen350/Labs.git**
      This is represented below:

```
PS C:\Users\Sujoy\Labs> git clone --bare https://github.tamu.edu/ecen350/Labs
Cloning into bare repository 'Labs.git'...
Username for 'https://github.tamu.edu': sujoy-saha
Password for 'https://sujoy-saha@github.tamu.edu':
remote: Counting objects: 76, done.
remote: Compressing objects: 100% (68/68), done.
remote: Total 76 (delta 8), reused 75 (delta 7), pack-reused 0
Unpacking objects: 100% (76/76), done.
Checking connectivity... done.
```

    - This will make a folder in your specified path on command-line called “Labs.git”.

- Change to Labs.git directory using **cd** command
- Push this into your newly created **private** repo as shown below:
    - Use method: **git push --mirror https://github.tamu.edu/NET-ID/ecen350lab.git**
    - **Note**: Use your github username(net-id in most cases) here. Git Bash also provides prompts for username and password

```
PS C:\Users\Sujoy\Labs> cd Labs.git
PS C:\Users\Sujoy\Labs\Labs.git> git push --mirror https://github.tamu.edu/sujoy-saha/ecen350lab.git
Username for 'https://github.tamu.edu': sujoy-saha
Password for 'https://sujoy-saha@github.tamu.edu':
Counting objects: 76, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (75/75), done.
Writing objects: 100% (76/76), 3.64 MiB | 2.04 MiB/s, done.
Total 76 (delta 8), reused 0 (delta 0)
To https://github.tamu.edu/sujoy-saha/ecen350lab.git
 * [new branch]      master -> master
```

- Go back to **previous directory** using **cd..**
- Delete Labs.git using:
    - rm -rf Labs.git (linux and Git Bash)
    - Remove-Item -Recurse -Force .\Labs.git\ (windows powershell)
- Download your private repo locally using clone method:
    - Use: git clone https://github.tamu.edu/**NET-ID**/ecen350lab.git
      You should get output like this

```
PS C:\Users\Sujoy> git clone https://github.tamu.edu/sujoy-saha/ecen350lab.git
Cloning into 'ecen350lab'...
Username for 'https://github.tamu.edu': sujoy-saha
Password for 'https://sujoy-saha@github.tamu.edu':
remote: Counting objects: 76, done.
remote: Compressing objects: 100% (67/67), done.
remote: Total 76 (delta 8), reused 76 (delta 8), pack-reused 0
Unpacking objects: 100% (76/76), done.
Checking connectivity... done.
```

- Your repo should look like this: (use ls command to show directory contents)

```
PS C:\Users\Sujoy> cd .\ecen350lab\
PS C:\Users\Sujoy\ecen350lab> ls


    Directory: C:\Users\Sujoy\ecen350lab


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        01-09-2016     15:30                lab1
d-----        01-09-2016     15:30                lab10
d-----        01-09-2016     15:30                lab2
d-----        01-09-2016     15:30                lab3
d-----        01-09-2016     15:30                lab4
d-----        01-09-2016     15:30                lab5
d-----        01-09-2016     15:30                lab6
d-----        01-09-2016     15:30                lab7
d-----        01-09-2016     15:30                lab8
d-----        01-09-2016     15:30                lab9
-a----        01-09-2016     15:30            395 .gitattributes
-a----        01-09-2016     15:30            740 .gitignore
```

Step 6 - Working with git

First, go to ecen350lab directory and go to a specific lab say **"lab1"**

- **STEP A**: Add files to Git to track changes:
  - For each file, enable tracking by Git using "add command"
  - Do: ***git add filename.extension***
    - ***git add lab1.s (example)***
  - Note you can add a folder as well. ***git add folder_name***
  - **[IMPORTANT]**, please remove any binary files like.bin, etc, which was not present originally in the repo. **Add all the files originally downloaded**. (**all .s files, .v files, .pdf files**). You can do this in the very beginning using **git add lab1** etc.
  - To add all the folders in the beginning(one time only): do this: **[IMPORTANT]**
  - Go to /ecen350lab/ directory and use: **git add ***
  - ***** denotes add everything(use only once before the first lab)**

```
Sujoy@DESKTOP-799MDOD MINGW64 ~/ecen350lab (master)
$ git add *
```

- **STEP B**: Edit and Commit:

  [IMPORTANT]: Please read **STEP D** and come back to **STEP B** if you have your **Git** repository in multiple machines because it is **very important** to pull any changes you have made before making any more changes and commit.

  - After making some changes like adding to the code/making edits in the file, make a commit on git. This includes pdf files.
  - A commit is kind of a record of change to a file
  - Command: **git commit -am "first edit"**

- “first edit” is a commit message. Can be anything.

- For submissions use “**submission**” as commit message **[IMPORTANT]**

```
PS C:\Users\Sujoy\Labs\lab2> notepad lab2a.asm
PS C:\Users\Sujoy\Labs\lab2> git add lab2a.asm
PS C:\Users\Sujoy\Labs\lab2> git commit -m "lab2a.asm"
[master e42f958] lab2a.asm
 1 file changed, 1 insertion(+), 1 deletion(-)
PS C:\Users\Sujoy\Labs\lab2>
```

- You can make as many edits as possible. **Make sure to add the file first** (if its not added already, else you will get the **following error**:)

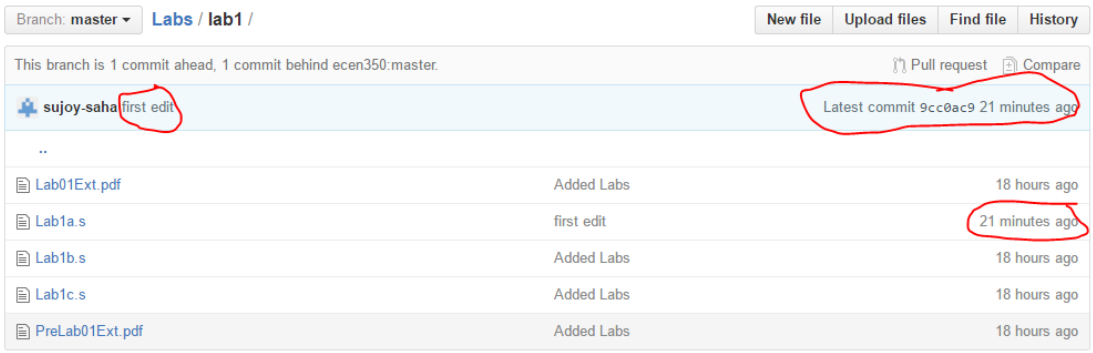
```
PS C:\Users\Sujoy\Labs\lab1> notepad Lab1a.s
PS C:\Users\Sujoy\Labs\lab1> git commit -m "first edit"
On branch master
Your branch is up-to-date with 'origin/master'.
Changes not staged for commit:
        modified:   Lab1a.s

no changes added to commit
```

- **STEP C**: **Push for Submissions**:
  - Git push is used to **upload data to your online Github account**.
  - **[IMPORTANT]** Please push your submission before your deadline. **We will only evaluate code pushed to your ONLINE repo before the specified deadline**
  - **Git Push Command: git push**
  - Please git commit first then git push. AS shown below

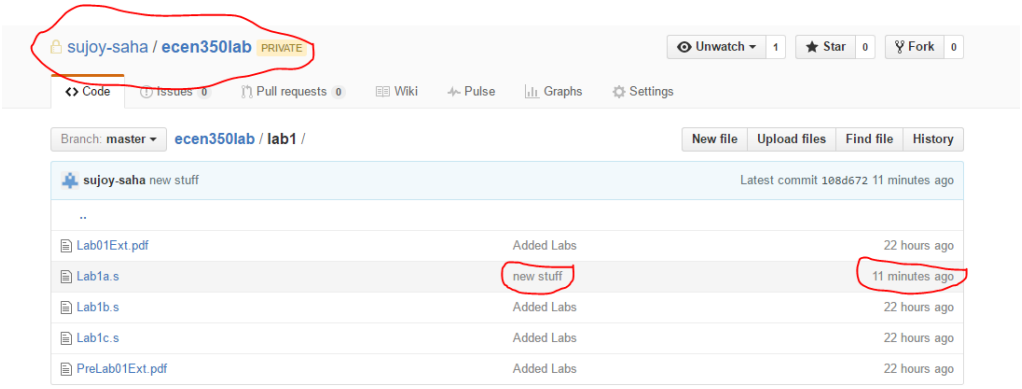
```
PS C:\Users\Sujoy\Labs> git commit -m "first edit"
[master 9cc0ac9] first edit
 1 file changed, 1 insertion(+), 1 deletion(-)
PS C:\Users\Sujoy\Labs> git push
warning: push.default is unset; its implicit value has changed in
Git 2.0 from 'matching' to 'simple'. To squelch this message
and maintain the traditional behavior, use:

  git config --global push.default matching

To squelch this message and adopt the new behavior now, use:

  git config --global push.default simple

When push.default is set to 'matching', git will push local branches
to the remote branches that already exist with the same name.

Since Git 2.0, Git defaults to the more conservative 'simple'
behavior, which only pushes the current branch to the corresponding
remote branch that 'git pull' uses to update the current branch.

See 'git help config' and search for 'push.default' for further information.
(the 'simple' mode was introduced in Git 1.7.11. Use the similar mode
'current' instead of 'simple' if you sometimes use older versions of Git)

Username for 'https://github.tamu.edu': [redacted]
Password for 'https://[redacted]@github.tamu.edu':
Counting objects: 4, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 344 bytes | 0 bytes/s, done.
Total 4 (delta 3), reused 0 (delta 0)
To https://github.tamu.edu/sujoy-saha/Labs.git
   aaabe3e..9cc0ac9  master -> master
```

- **Note: Commit many times, push fewer times.**
- **Note: you will be required to add username and password while pushing/cloning (use NET-ID and password).**

- **After committing and pushing your repos should be updated like this:**



- **[IMPORTANT]** Commit messages **(example-> "new stuff")** are shown below, your final submission should have comment "**submission**".
- You can overwrite previous commits.



- **Finally it should look like the above.**

- **STEP D**: **Clone** once and **Pull at any time before** you start working **on your home machine**

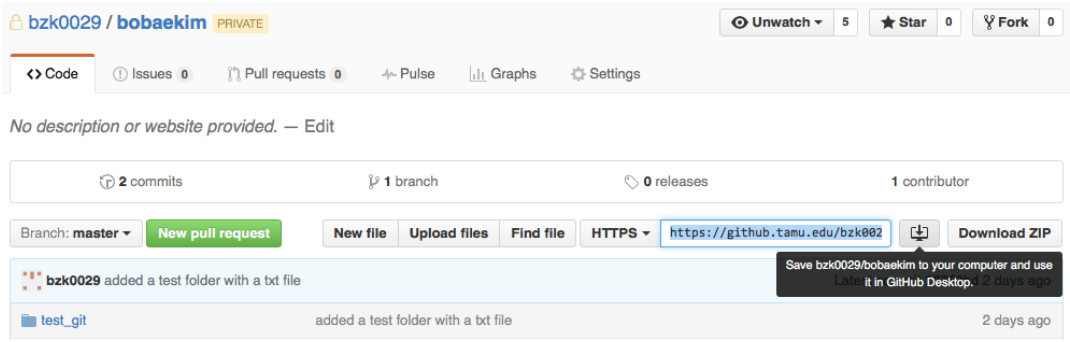
  - First of all, make sure **Git** is installed on your home machine
    (see Step 4 of this manual or click on the link below for further details:
    https://git-scm.com/book/en/v2/Getting-Started-Installing-Git)

  - Once **Git** is installed, open Terminal. Change the current working directory to the location where you want your directory to be made.

  - **Note**: You will copy an existing **Git** repository by using the **'git clone'** command. In the example shown below, I will copy my existing **Git** repository to Desktop.

  - **Note**: You won't need to repeat this step for the same **Git** repository. In other words, this **'git clone'** step is an one-time thing you would need to do for your private repository (unless you have multiple repositories you would like to copy and locate to the directory on your home machine).

  - Go to https://github.tamu.edu/YOUR-USERNAME/YOUR-REPOSITORY
  - Copy the URL for your **Git** repository as shown in the picture below:



  - Type the **'git clone'** command to terminal window

> git clone [URL for your private **Git** repository.git]

`git clone https://github.tamu.edu/YOUR-USERNAME/YOUR-REPOSITORY`

- For example, I can copy my private **Git** repository using following command:
  > `git clone https://github.tamu.edu/bzk0029/bobaekim.git`

```
Bobaes-Air% cd Desktop
Bobaes-Air% git clone https://github.tamu.edu/bzk0029/bobaekim.git
Cloning into 'bobaekim'...
remote: Counting objects: 7, done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 7 (delta 0), reused 4 (delta 0), pack-reused 0
Unpacking objects: 100% (7/7), done.
Checking connectivity... done.
```

Now, I have copied my private **Git** repository to Desktop (replace this to the directory you would like to locate your cloned **Git** repository) on my Macbook Air.

- Once you locate your **Git** repository to the directory on your home machine, you can edit and commit by following **STEP B** under **Step 6** of this document.
- **[IMPORTANT]**: At any time, You **must** use the 'git pull' command to make sure that your **Git** repository on your home machine or Linux machine in the lab is up-to-date before you make any changes to the repository.

- You will copy an existing **Git** repository by using the **'git pull'** command.
- Go to the directory where your **Git** repository is located.
- For example, I've changed the current working directory to the directory, which is equivalent to the cloned **Git** repository, */Users/bobaekim/Desktop/bobaekim*

- In your **Git** repository, simply type the **'git pull'** command to terminal window
- Note that I chose not to make any changes here. Therefore, we will see nothing but a 'Already up-to-date' message when we type the command to pull any changes made to my **Git** repository. In your case, you will see what changes have been made to your **Git** repository.

> `git pull`

```
Bobaes-Air% cd bobaekim
Bobaes-Air% git pull
Already up-to-date.
```

- Once you pull all the changes that you made to your **Git** repository (on the same or different machine), it's now ready for you to make other changes and commit.