Assignment 01

Program Name	BSCS
Course Code	CSSE3143
Course Name	Web Application Development
Course Instructor	Muhammad Ali Makhdoom
Time Allowed	**Until 29th Oct, 2018**
Assessment	VCS (Git & Github), CMD
Special Instructions	1. Do not share your solution with other students. 2. Submit the softcopy of your solution on or before the due date. Late submission will result in some penalty. 3. Always keep a backup of your solution till it is graded.
Conditions	This is an individual assignment
Total marks available	100
Academic Honesty Policy	Academic dishonesty will not be tolerated. Academic dishonesty includes cheating, plagiarism (copying) or any other attempt to gain an academic advantage in a dishonest or unfair manner.

Introduction

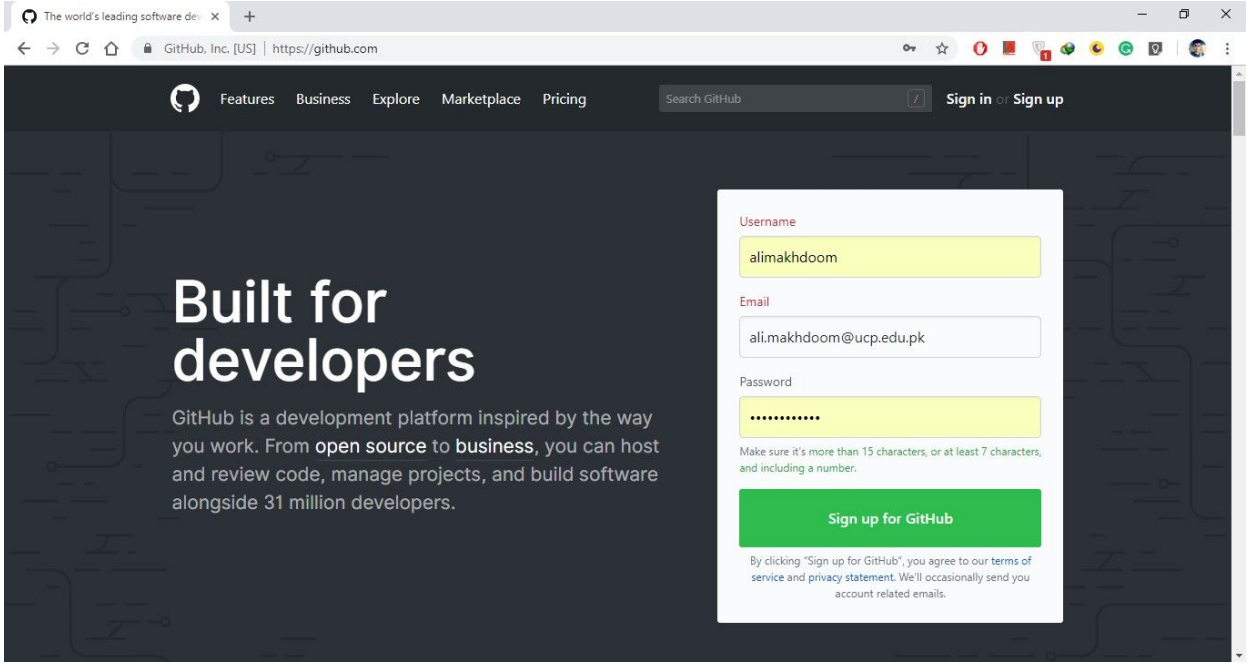
This assignment is designed to help you become familiar with the Git version control system that you will be using to manage the source code of your projects. Git is a distributed version control system designed to be fast and simple. You can think like versions (or snapshots) of your project are saved (committed) over time and you can freely switch between them. Once you commit a snapshot, it is very difficult to lose it, especially if you constantly push your repository to a save storage (such as GitHub). So, you can experiment without danger of screwing up your project severely.

In this assignment, we will learn basic commands and workflows for Git and GitHub. There are many in-depth Git tutorials online, e.g. https://git-scm.com/book/en/v2 (many things in this tutorial are from this source)

1. **Create a GitHub Account**
2. **Create a test_git_*registration_number* Repository**
3. **Install Git**
4. **Clone Your GitHub Repository To Your Local Machine**
5. **Basic Workflow**
6. **Pulling Changes**
7. **Merge Conflicts**
8. **Branching**
   - I. **Create Branches**
   - II. **Switch Branch**
   - III. **Working and Merging Branches**
9. **Submit Your Assignments**
   - I. **Upload a ZIP file**
   - II. **Submit on LMS**
10. **Further Readings**

1. Create a GitHub Account

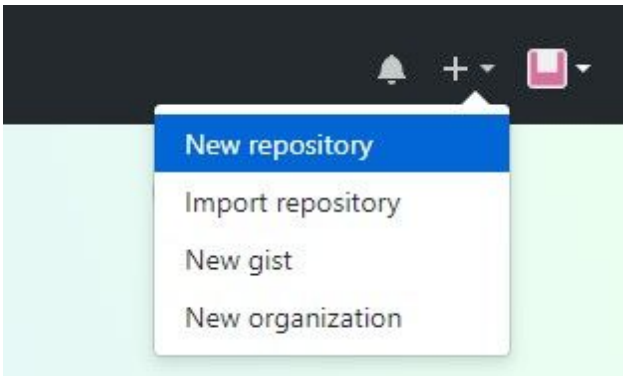
Throughout the course, we will use GitHub as the remote server for storing our repositories. So, if you do not have a GitHub account, go to https://github.com/ and create one. You will need to select a username and email address when you register. Select a professional username according to your real registered name in the university and avoid selecting a random name like “dad’s princess” etc.



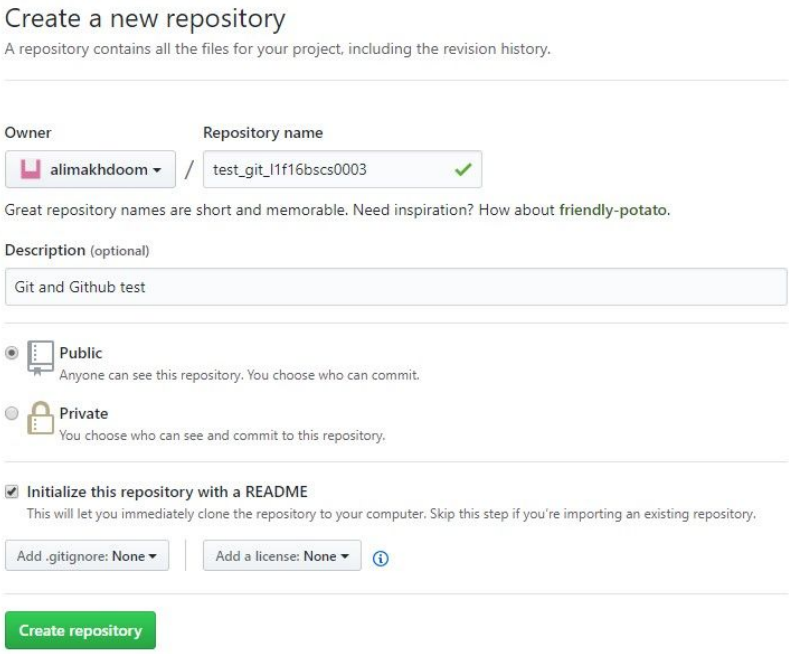
Set a profile picture for your account by clicking your account name in the upper left-hand corner, and then clicking on the giant picture. It will simply make reading the commit history much easier for all collaborators when working on the project. Once you have it, it may take 15 minutes for it to take effect, but if you clear the cache in your browser it may be instant.

2. Create a test_git_*registeration_number* Repository

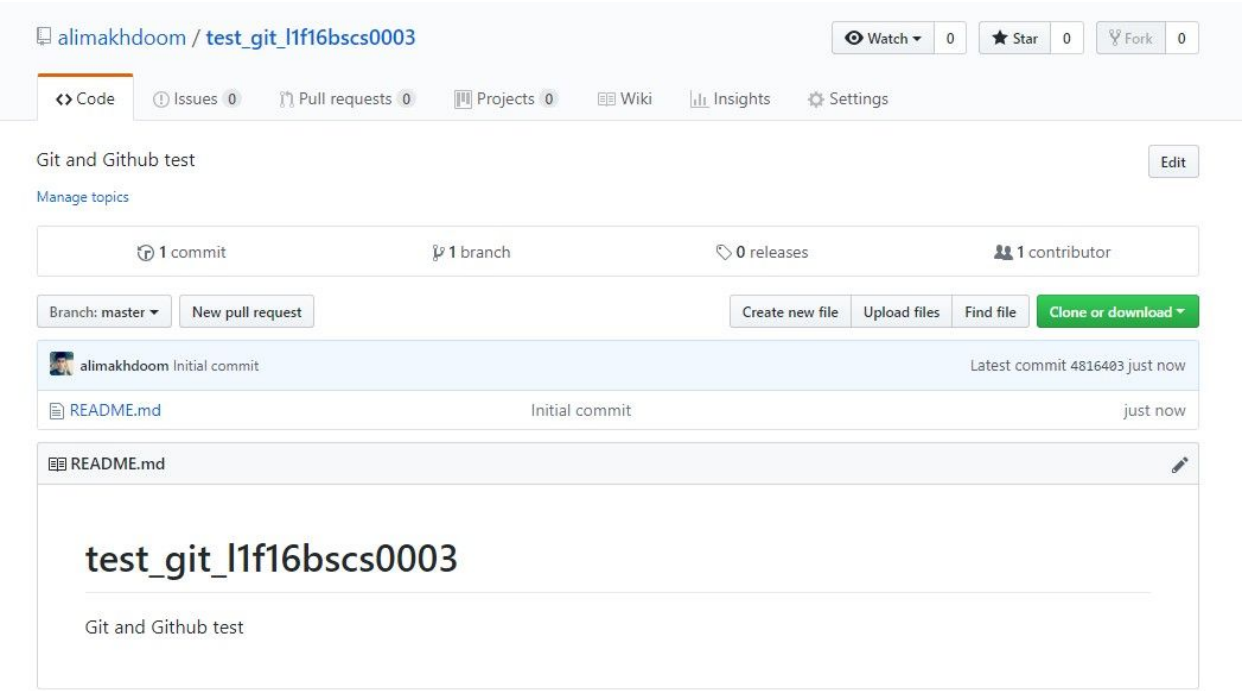
- Click on "Create new..." button (+ sign) and select "New repository"



- Fill in the information in the next page (the following information is just a suggestion):
  - Repository name: test_git_*registration_number*
  - Description: Git and Github test
  - Choose Public
  - Choose "Initialize this repository with a README" (this will create a README file for you)

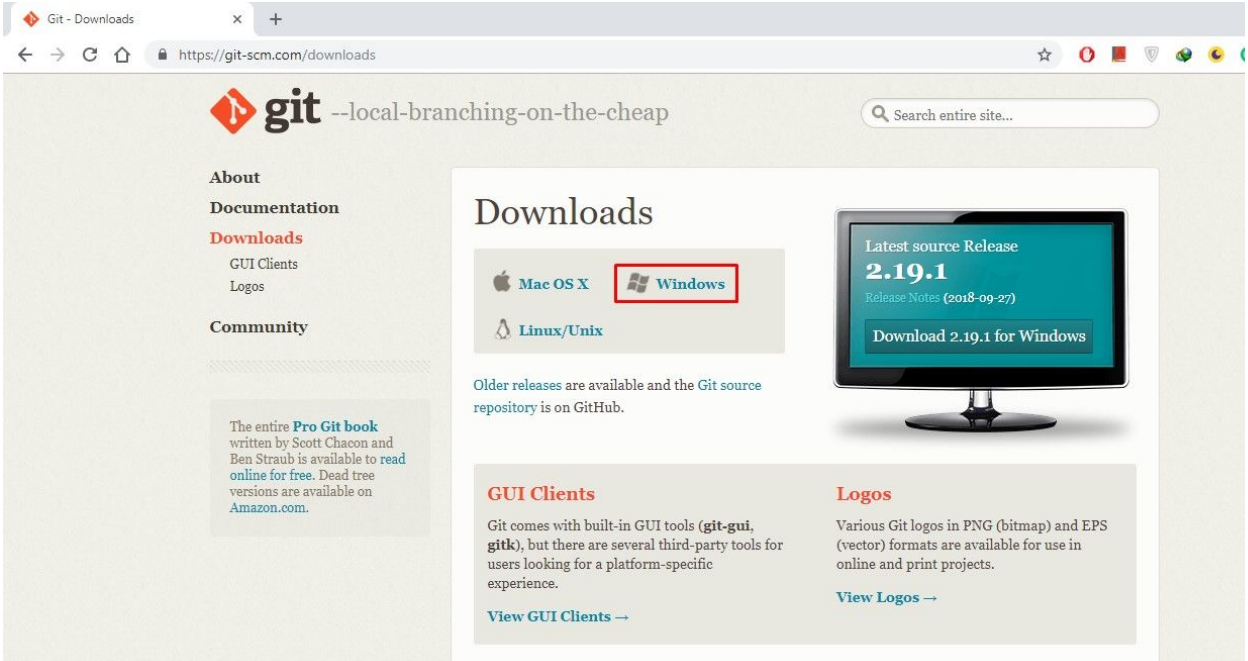


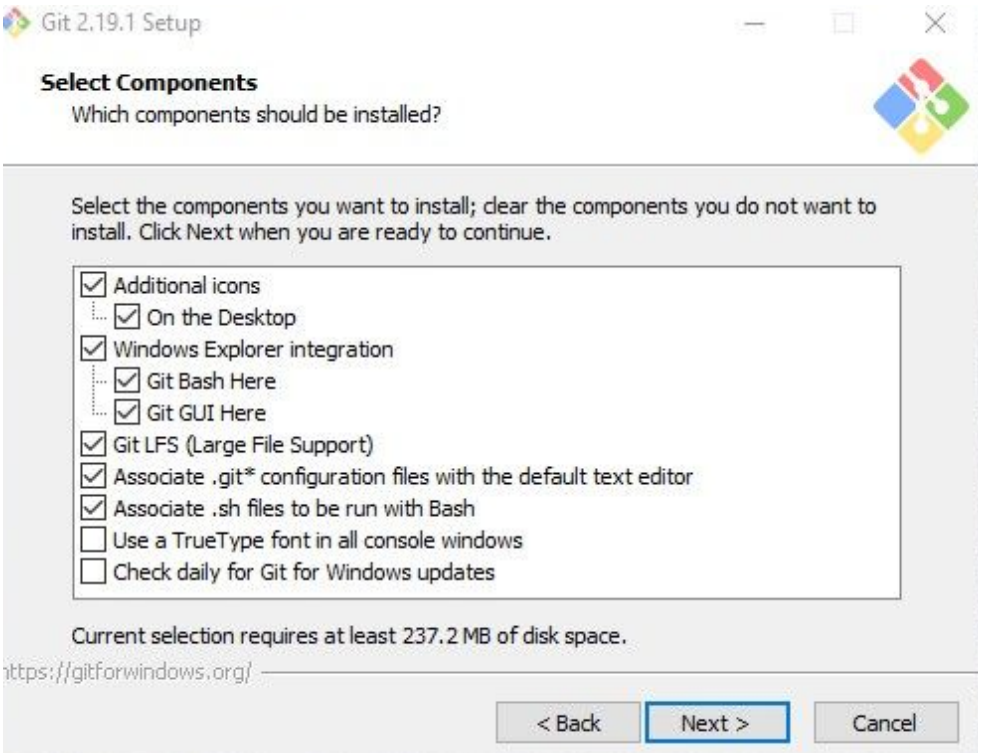
- After clicking "Create repository", if success, you will see your new repository



3. Install Git

- Make sure that git is installed in your system
  - https://git-scm.com/book/en/v2/Getting-Started-Installing-Git


Git - Downloads
https://git-scm.com/downloads
git --local-branching-on-the-cheap
Search entire site...
About
Documentation
Downloads
GUI Clients
Logos
Community
The entire Pro Git book written by Scott Chacon and Ben Straub is available to read online for free. Dead tree versions are available on Amazon.com.
Downloads
Mac OS X
Windows
Linux/Unix
Older releases are available and the Git source repository is on GitHub.
Latest source Release
2.19.1
Release Notes (2018-09-27)
Download 2.19.1 for Windows
GUI Clients
Git comes with built-in GUI tools (git-gui, gitk), but there are several third-party tools for users looking for a platform-specific experience.
View GUI Clients →
Logos
Various Git logos in PNG (bitmap) and EPS (vector) formats are available for use in online and print projects.
View Logos →


Git 2.19.1 Setup
Select Components
Which components should be installed?
Select the components you want to install; clear the components you do not want to install. Click Next when you are ready to continue.
Additional icons
On the Desktop
Windows Explorer integration
Git Bash Here
Git GUI Here
Git LFS (Large File Support)
Associate .git* configuration files with the default text editor
Associate .sh files to be run with Bash
Use a TrueType font in all console windows
Check daily for Git for Windows updates
Current selection requires at least 237.2 MB of disk space.
https://gitforwindows.org/
< Back
Next >
Cancel

Git 2.19.1 Setup

Adjusting your PATH environment

How would you like to use Git from the command line?

○ **Use Git from Git Bash only**

This is the safest choice as your PATH will not be modified at all. You will only be able to use the Git command line tools from Git Bash.

◉ **Use Git from the Windows Command Prompt**

This option is considered safe as it only adds some minimal Git wrappers to your PATH to avoid cluttering your environment with optional Unix tools. You will be able to use Git from both Git Bash and the Windows Command Prompt.

○ **Use Git and optional Unix tools from the Windows Command Prompt**

Both Git and the optional Unix tools will be added to your PATH.
Warning: This will override Windows tools like "find" and "sort". Only use this option if you understand the implications.

https://gitforwindows.org/

< Back | Next > | Cancel

Git 2.19.1 Setup

Choosing HTTPS transport backend

Which SSL/TLS library would you like Git to use for HTTPS connections?

○ **Use the OpenSSL library**

Server certificates will be validated using the ca-bundle.crt file.

◉ **Use the native Windows Secure Channel library**

Server certificates will be validated using Windows Certificate Stores.
This option also allows you to use your company's internal Root CA certificates distributed e.g. via Active Directory Domain Services.

https://gitforwindows.org/

< Back | Next > | Cancel

Git 2.19.1 Setup

Configuring the line ending conversions

How should Git treat line endings in text files?

- (●) **Checkout Windows-style, commit Unix-style line endings**

  Git will convert LF to CRLF when checking out text files. When committing text files, CRLF will be converted to LF. For cross-platform projects, this is the recommended setting on Windows ("core.autocrlf" is set to "true").

- ( ) **Checkout as-is, commit Unix-style line endings**

  Git will not perform any conversion when checking out text files. When committing text files, CRLF will be converted to LF. For cross-platform projects, this is the recommended setting on Unix ("core.autocrlf" is set to "input").

- ( ) **Checkout as-is, commit as-is**

  Git will not perform any conversions when checking out or committing text files. Choosing this option is not recommended for cross-platform projects ("core.autocrlf" is set to "false").

https://gitforwindows.org/

< Back | Next > | Cancel

Git 2.19.1 Setup

Configuring the terminal emulator to use with Git Bash

Which terminal emulator do you want to use with your Git Bash?

- ( ) **Use MinTTY (the default terminal of MSYS2)**

  Git Bash will use MinTTY as terminal emulator, which sports a resizable window, non-rectangular selections and a Unicode font. Windows console programs (such as interactive Python) must be launched via `winpty` to work in MinTTY.

- (●) **Use Windows' default console window**

  Git will use the default console window of Windows ("cmd.exe"), which works well with Win32 console programs such as interactive Python or node.js, but has a very limited default scroll-back, needs to be configured to use a Unicode font in order to display non-ASCII characters correctly, and prior to Windows 10 its window was not freely resizable and it only allowed rectangular text selections.

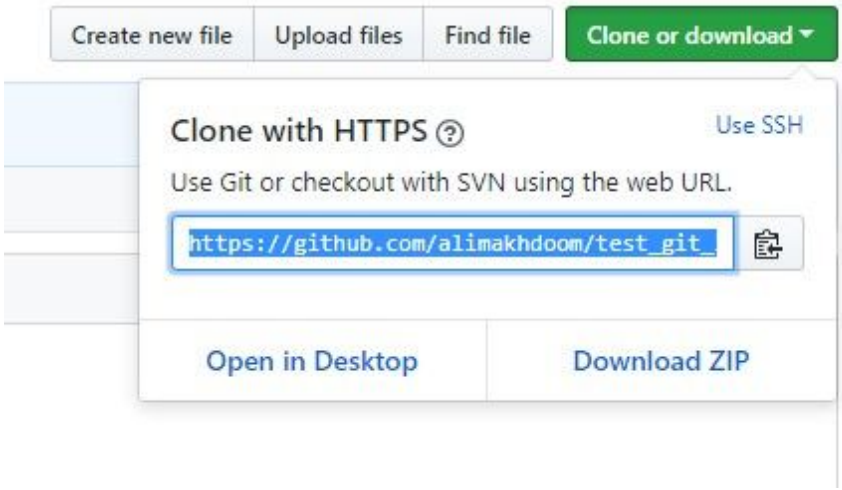
https://gitforwindows.org/

< Back | Next > | Cancel

```
Command Prompt

C:\Users\Ali Makhdoom>git config --global user.name "Muhammad Ali Makhdoom"

C:\Users\Ali Makhdoom>git config --global user.email "14muhammad@gmail.com"

C:\Users\Ali Makhdoom>git config --list
core.symlinks=false
core.autocrlf=true
core.fscache=true
color.diff=auto
color.status=auto
color.branch=auto
color.interactive=true
help.format=html
rebase.autosquash=true
http.sslbackend=schannel
diff.astextplain.textconv=astextplain
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge -- %f
filter.lfs.process=git-lfs filter-process
filter.lfs.required=true
credential.helper=manager
core.editor='C:\Program Files (x86)\Notepad++\notepad++.exe' -multiInst -notabbar -nosession -noPlugin
user.name=Muhammad Ali Makhdoom
user.email=14muhammad@gmail.com

C:\Users\Ali Makhdoom>
```

4. Clone Your GitHub Repository To Your Local Machine

- Go to your GitHub repository, click on "Clone or download" button and copy the link to your repository



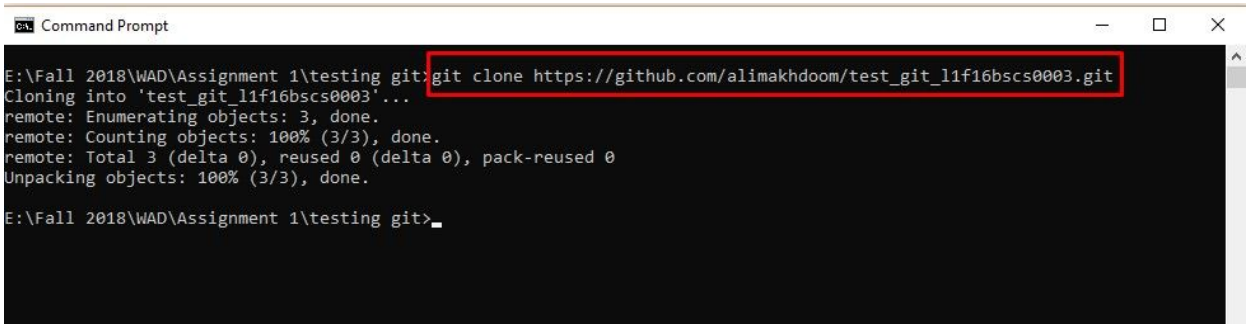
- Open your terminal application, change directory to folder you want to save your repository, then clone your repository by:

  `git clone <link_to_your_repository_on_GitHub>`

e.g. in my case:

```
git clone https://github.com/alimakhdoom/test_git_l1f16bscs0003.git
```

- Git will clone your repository to a new folder with the same name.



- If you go into the test_git_*registration_number* folder, you can see the content of your repository, along with .git folder created by Git. (In macOS and Linux, .git and .gitignore are hidden by default)

5. Basic Workflow

In Git, there are 3 main states your files can be in *modified*, *staged*, *committed*.

- A file is in the ***modified*** state if it is modified but has not been committed to the database of your repository.
- A modified file is in the ***staged*** state if its current version is marked to go into your next commit snapshot (using commands like **add, mv, reset, rm**)
- A file is in a ***committed*** state if it is safely stored in the database of your repository (using **commit** command)

In this course, since we use GitHub to store our repository remotely, you can consider another state, called ***pushed*** if a committed file is stored in a remote repository (using push command).
At any time, you can check the status of your files using **git status** command

Thus, a basic workflow for modifying a file (e.g. myinfo.txt) can be:

- Modify the file (e.g.):

```
echo "Name: Muhammad Ali Makhdoom" >> myinfo.txt
```

- Stage the changes:

```
git add myinfo.txt
```

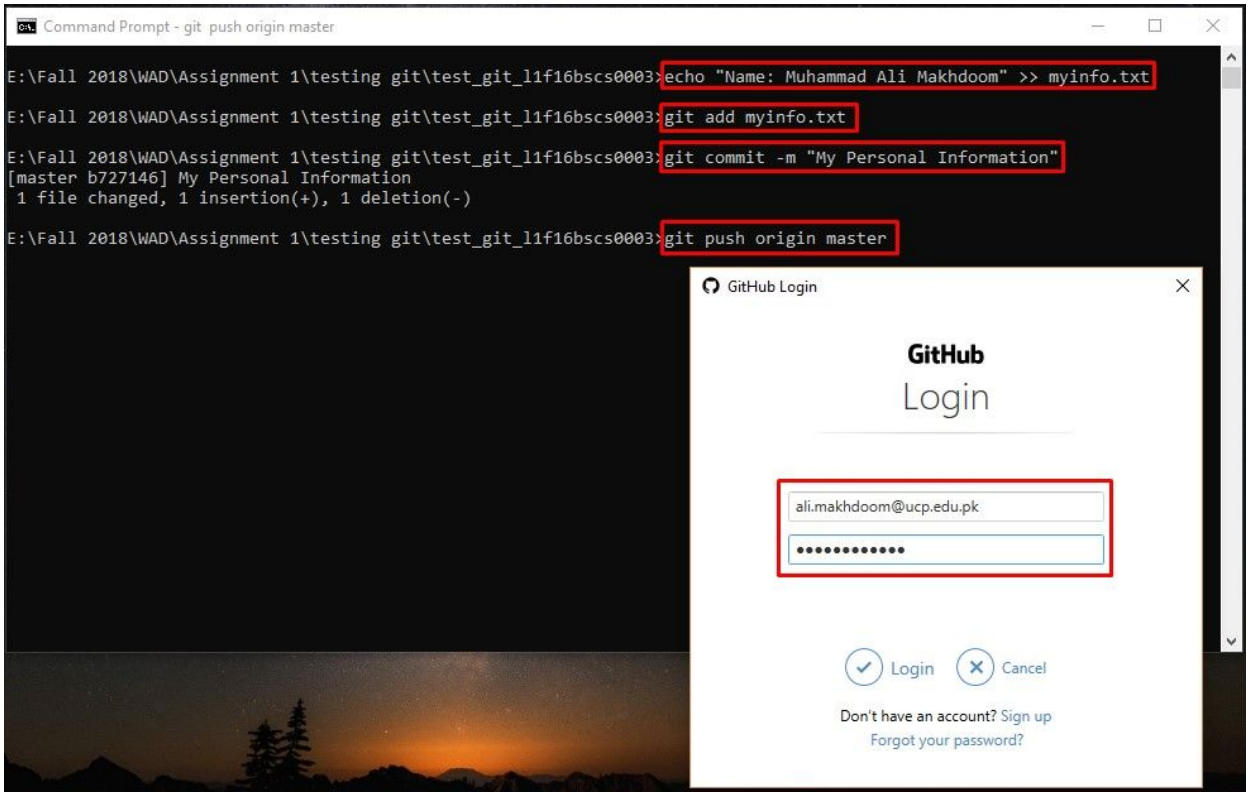
- Commit the changes:

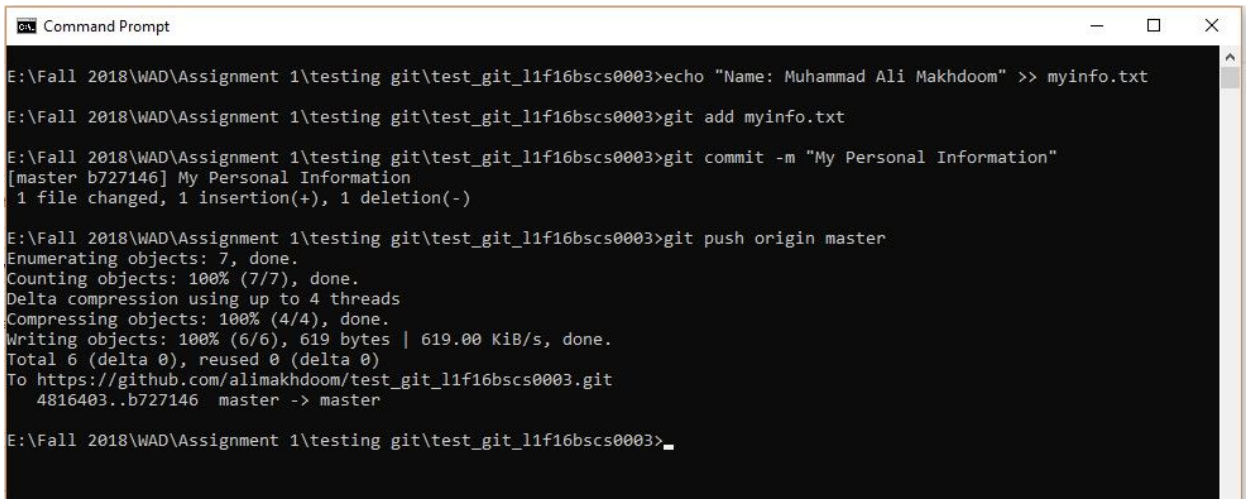
```
git commit -m "My Personal Information"
```

- Push commit to GitHub:

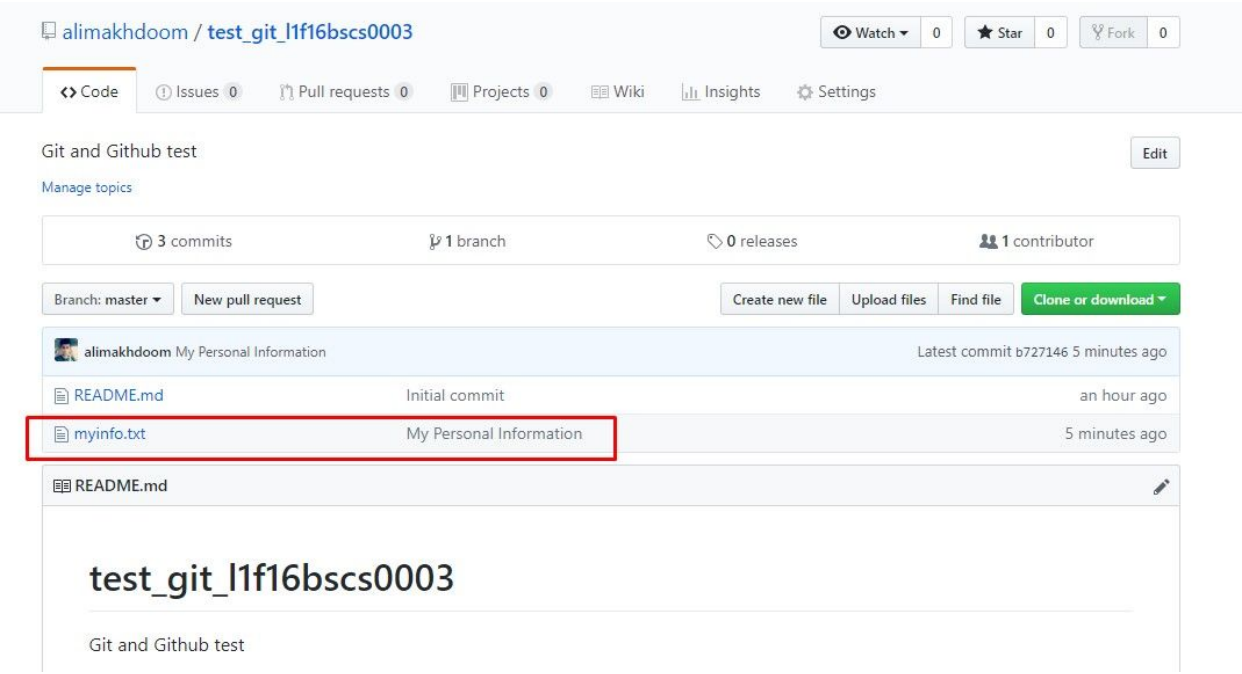
```
git push origin master
```

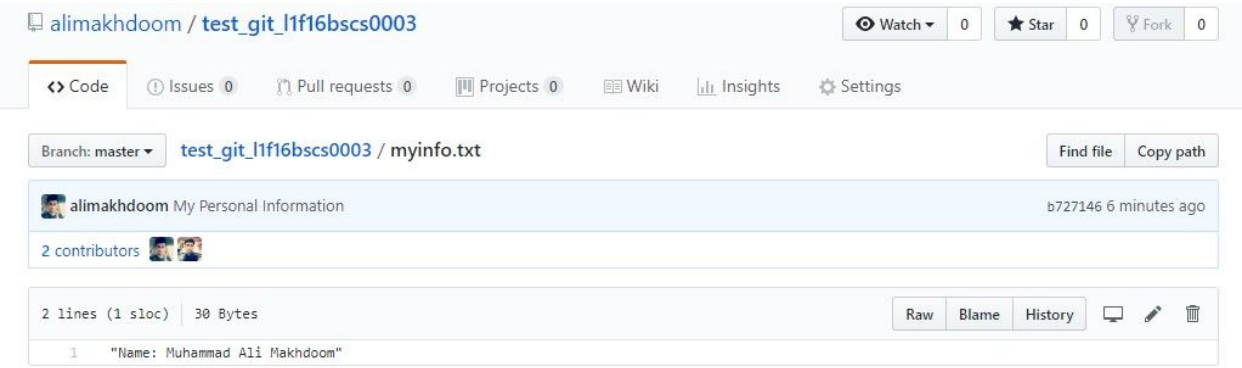
Here are screenshots for this workflow:





If you go to your repository on GitHub you can see myinfo.txt, along with the commit message.



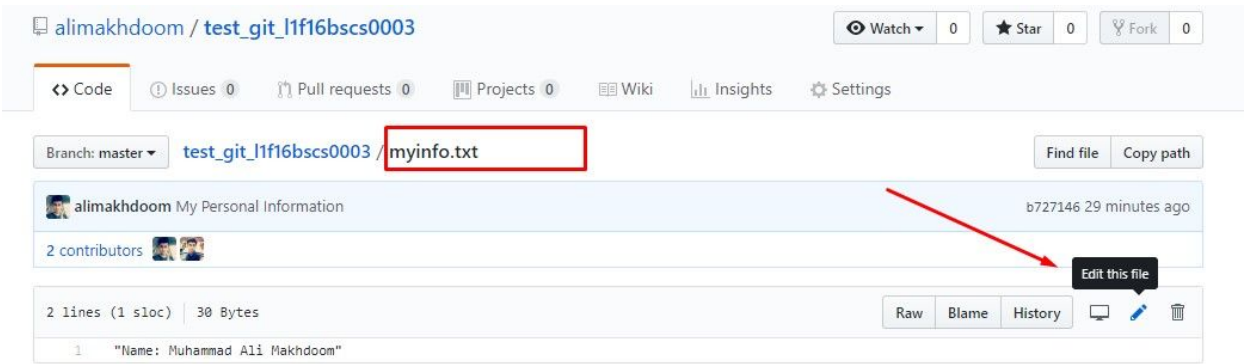


So, you can remember: **add** - **commit** - **push**
Remember, anything that is committed in Git can almost always be recovered. However, anything you lose that was never committed is likely never to be seen again. If you want to revert to a commit (e.g. because of some faulty commits), you can use **git revert <commit>**.

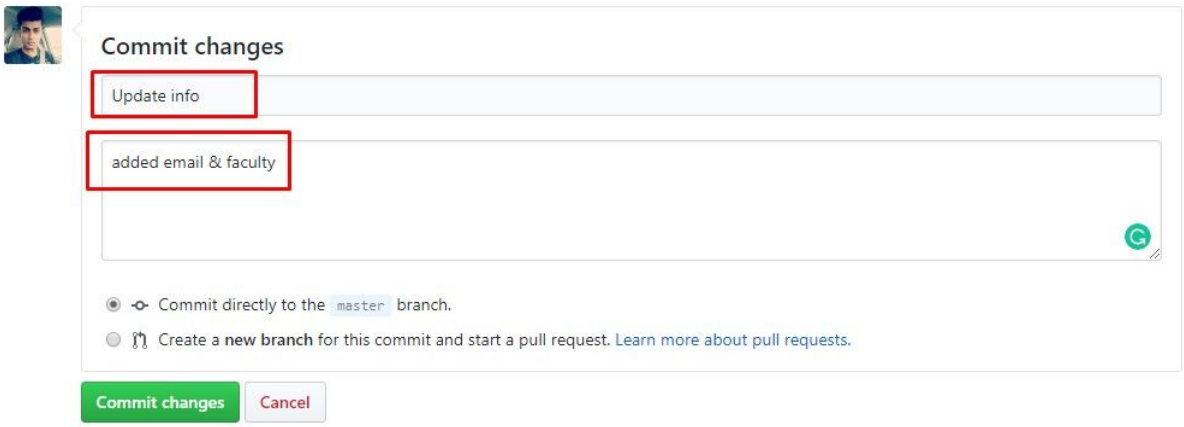
6. Pulling Changes

Very often, your remote repository is changed (e.g. by you or your teammates) and you need to update your local repository accordingly. You can do that using pull command (You should commit your local changes first). Here, we will edit "myinfo.txt" file on GitHub, then pull the change to our local repository.

- Click on "myinfo.txt" file in your repository on GitHub, then click "Edit this file" icon.



- Edit the content of the file and commit changes



- When you go back to your repository on GitHub, the content of "myinfo.txt" and commit message will be updated
- Go back to your terminal and pull the change
    - git pull

```
E:\Fall 2018\WAD\Assignment 1\testing git\test_git_l1f16bscs0003>git pull
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
From https://github.com/alimakhdoom/test_git_l1f16bscs0003
   b727146..b8c7229  master     -> origin/master
Updating b727146..b8c7229
Fast-forward
 myinfo.txt | 2 ++
 1 file changed, 2 insertions(+)

E:\Fall 2018\WAD\Assignment 1\testing git\test_git_l1f16bscs0003>
```

Or you can run:

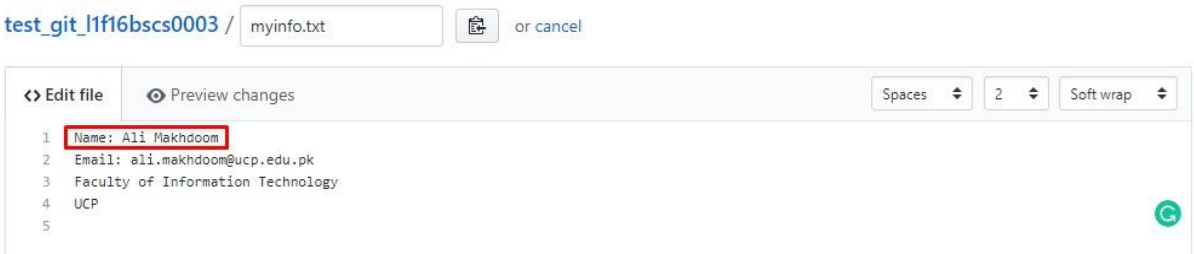
- **git fetch origin**
- **git merge**

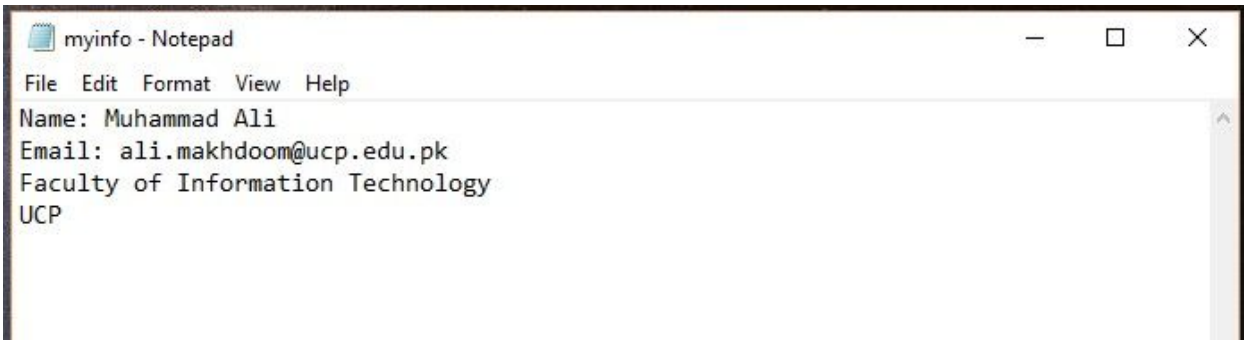
```
E:\Fall 2018\WAD\Assignment 1\testing git\test_git_l1f16bscs0003>git fetch origin
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
From https://github.com/alimakhdoom/test_git_l1f16bscs0003
   b8c7229..745d8f9  master     -> origin/master

E:\Fall 2018\WAD\Assignment 1\testing git\test_git_l1f16bscs0003>git merge
Updating b8c7229..745d8f9
Fast-forward
 myinfo.txt | 1 +
 1 file changed, 1 insertion(+)

E:\Fall 2018\WAD\Assignment 1\testing git\test_git_l1f16bscs0003>
```

- The content of "myinfo.txt" should be changed.

7. Merge Conflicts

Sometimes you may have conflicts when you merge the same code block that was modified in different repositories. For example, on GitHub, modify the name in your "myinfo.txt", then commit. Do the same thing on your local machine: modify in your "myinfo.txt", then commit.






```
E:\Fall 2018\WAD\Assignment 1\testing git\test_git_l1f16bscs0003>git add myinfo.txt

E:\Fall 2018\WAD\Assignment 1\testing git\test_git_l1f16bscs0003>git commit -m "Name is updated"
[master ab36430] Name is updated
 1 file changed, 1 insertion(+), 1 deletion(-)

E:\Fall 2018\WAD\Assignment 1\testing git\test_git_l1f16bscs0003>git push origin master
To https://github.com/alimakhdoom/test_git_l1f16bscs0003.git
 ! [rejected]        master -> master (fetch first)
error: failed to push some refs to 'https://github.com/alimakhdoom/test_git_l1f16bscs0003.git'
hint: Updates were rejected because the remote contains work that you do
hint: not have locally. This is usually caused by another repository pushing
hint: to the same ref. You may want to first integrate the remote changes
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.

E:\Fall 2018\WAD\Assignment 1\testing git\test_git_l1f16bscs0003>_
```

After changing the content of "myinfo.txt", you can add and commit with:

- **git add myinfo.txt**
- **git commit -m "Update myinfo"**

Now, on your local machine, if you pull the remote repository, you will get an error message saying that you have conflicts in "myinfo.txt" and you need to fix them, then commit the result.

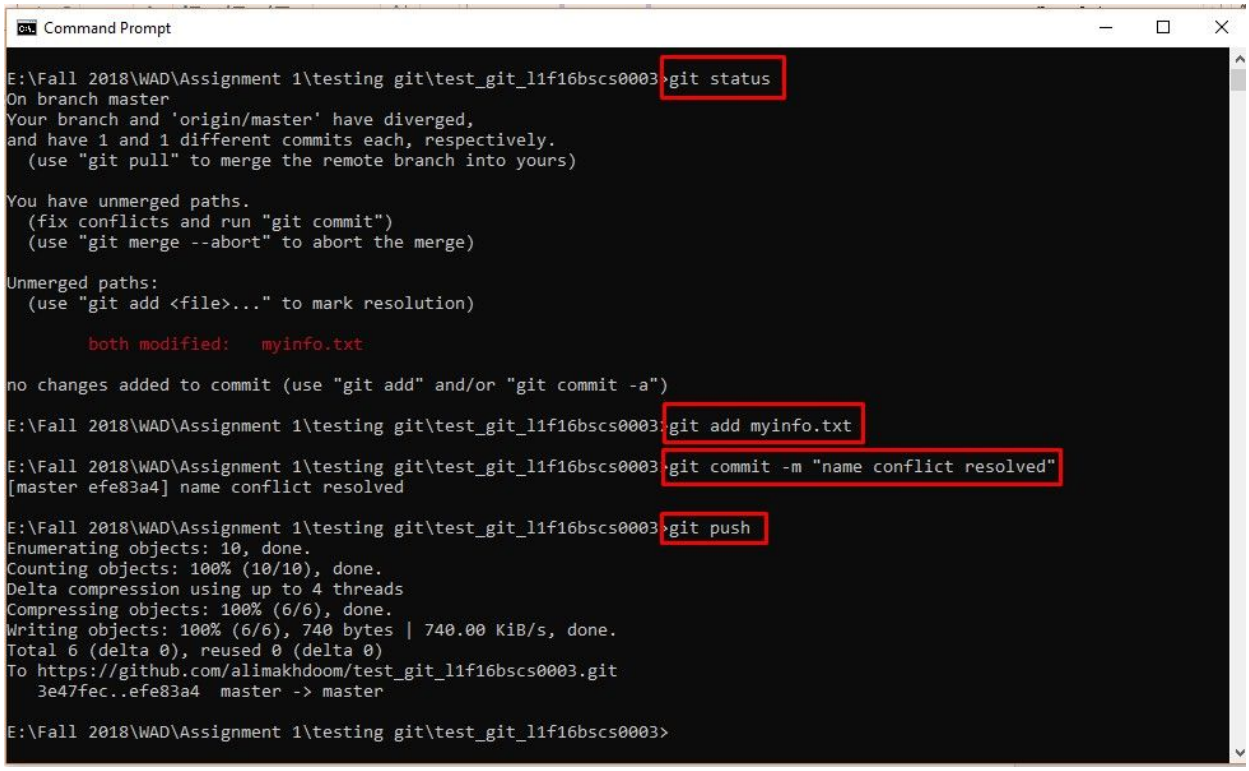
```
E:\Fall 2018\WAD\Assignment 1\testing git\test_git_l1f16bscs0003>git pull
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
From https://github.com/alimakhdoom/test_git_l1f16bscs0003
   745d8f9..3e47fec  master     -> origin/master
Auto-merging myinfo.txt
CONFLICT (content): Merge conflict in myinfo.txt
Automatic merge failed; fix conflicts and then commit the result.

E:\Fall 2018\WAD\Assignment 1\testing git\test_git_l1f16bscs0003>
```

When you open "myinfo.txt" file on your local machine, you will see something like this

```
myinfo - Notepad
File  Edit  Format  View  Help
<<<<<<< HEAD
Name: Muhammad Ali
=======
Name: Ali Makhdoom
>>>>>>> 3e47fec6fb556d12a2be69b3c4ce9619384d74ce
Email: ali.makhdoom@ucp.edu.pk
Faculty of Information Technology
UCP
```

" <<<<<<<" and " >>>>>>>" denote a conflict block. Content before " =======" is on your local machine, and content after " ======= " is on the remote repository (GitHub).
You need to resolve this, delete lines with " <<<<<<<" and " >>>>>>>", add and commit the change, then push to the remote repository. For example, we can change to:

```
myinfo - Notepad
File  Edit  Format  View  Help
Name: Muhammad Ali
=======
Name: Ali Makhdoom
Email: ali.makhdoom@ucp.edu.pk
Faculty of Information Technology
UCP
```



After changing the content of "myinfo.txt", you can add and commit with:

- **git add myinfo.txt**
- **git commit -m "name conflict resolved"**
- **git push**

8. Branching

Let's say that your program is now stable and you have released it to your customers. You want to add more features to it but do not want to mess up the stable version. If you do not use branches (i.e. just work on your main branch), your main repository will keep adding new updates, bugs, fixes, tests and moving between commits.

A branch can be considered as a development line. You can create a new branch, work in that line without worrying about screwing up the main (stable) line, then merge to the main

line when you are ready. You can look at more in-depth information about branching here: https://git- scm.com/book/en/v2/Git-Branching-Branches-in-a-Nutshell
When you create a new repository, Git automatically creates a branch "master". Because it is everywhere (and most people do not bother to change it), it is often considered as the main (stable) branch.

I. Create Branches

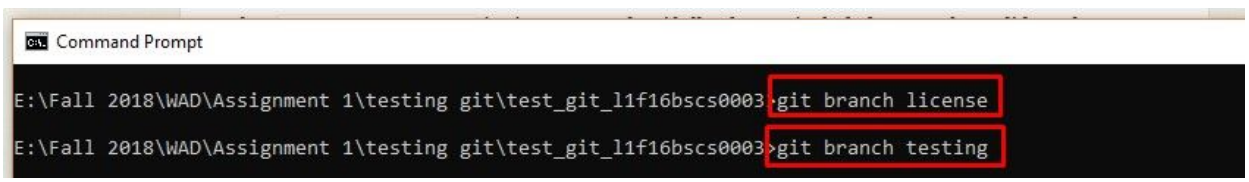
You can create a branch with the **git branch** command. Let's create 2 branches: *license* and *testing*.

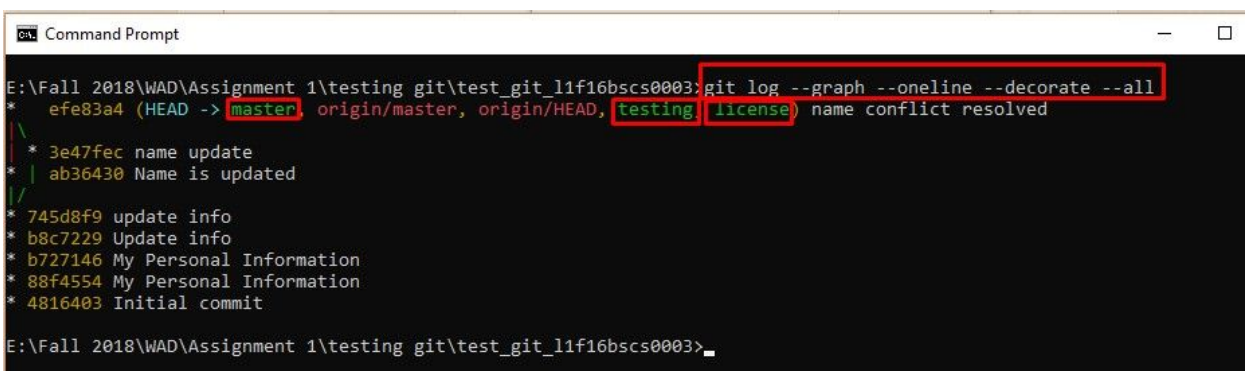
- **git branch license**
- **git branch testing**

To see your branches and which branch you are working on, you use:

- **git log --graph --oneline --decorate --all**

We have 3 branches (*master*, *testing*, *license* - you can ignore *origin/master* and *origin/HEAD* for now). The HEAD pointer indicates that we are at the *master* branch.





II. Switch Branch

You can switch to another branch with the **git checkout <branch_name>** command.

- **git checkout testing**
- **git log --graph --oneline --decorate --all**

```
Command Prompt

E:\Fall 2018\WAD\Assignment 1\testing git\test_git_l1f16bscs0003>git checkout testing
Switched to branch 'testing'

E:\Fall 2018\WAD\Assignment 1\testing git\test_git_l1f16bscs0003>git log --graph --oneline --decorate --all
*   efe83a4 (HEAD -> testing, origin/master, origin/HEAD, master, license) name conflict resolved
|\
| * 3e47fec name update
* | ab36430 Name is updated
|/
* 745d8f9 update info
* b8c7229 Update info
* b727146 My Personal Information
* 88f4554 My Personal Information
* 4816403 Initial commit

E:\Fall 2018\WAD\Assignment 1\testing git\test_git_l1f16bscs0003>
```

III. Working and Merging Branches

Let's create a "test.txt" file in *testing* branch, switch to *license* branch and create a "LICENSE" file there, switch back to the *master* branch, merge other branches to the master branch, and delete other branches.

- Create a "test.txt" file in the testing branch

```
Command Prompt

E:\Fall 2018\WAD\Assignment 1\testing git\test_git_l1f16bscs0003>echo "just a text file" >> test.txt

E:\Fall 2018\WAD\Assignment 1\testing git\test_git_l1f16bscs0003>
```

- Commit the change and check status. You can use **git log** to see more details about branches.
    - **git status**
    - **git add test.txt**
    - **git commit -m "test.txt file added"**
    - **git log --graph --oneline --decorate --all**

```
Command Prompt

E:\Fall 2018\WAD\Assignment 1\testing git\test_git_l1f16bscs0003>echo "just a text file" >> test.txt

E:\Fall 2018\WAD\Assignment 1\testing git\test_git_l1f16bscs0003>git status
On branch testing
Untracked files:
  (use "git add <file>..." to include in what will be committed)

        test.txt

nothing added to commit but untracked files present (use "git add" to track)

E:\Fall 2018\WAD\Assignment 1\testing git\test_git_l1f16bscs0003>git add test.txt

E:\Fall 2018\WAD\Assignment 1\testing git\test_git_l1f16bscs0003>git commit -m "test.txt file added"
[testing 6a4a3ea] test.txt file added
 1 file changed, 1 insertion(+)
 create mode 100644 test.txt

E:\Fall 2018\WAD\Assignment 1\testing git\test_git_l1f16bscs0003>git log --graph --oneline --decorate --all
* 6a4a3ea (HEAD -> testing) test.txt file added
*   efe83a4 (origin/master, origin/HEAD, master, license) name conflict resolved
|\
| * 3e47fec name update
* | ab36430 Name is updated
|/
* 745d8f9 update info
* b8c7229 Update info
* b727146 My Personal Information
* 88f4554 My Personal Information
* 4816403 Initial commit

E:\Fall 2018\WAD\Assignment 1\testing git\test_git_l1f16bscs0003>_
```

- Now, switch to the *license* branch. You will not see "test.txt" there because it is in the *testing* branch, not the *license* branch.
    - **git checkout license**

```
Command Prompt

E:\Fall 2018\WAD\Assignment 1\testing git\test_git_l1f16bscs0003>git status
On branch testing
nothing to commit, working tree clean

E:\Fall 2018\WAD\Assignment 1\testing git\test_git_l1f16bscs0003>git checkout license
Switched to branch 'license'

E:\Fall 2018\WAD\Assignment 1\testing git\test_git_l1f16bscs0003>_
```

- Create a "LICENSE" file, add to git, commit the change, and check status: After creating LICENSE file:
    - **git add LICENSE**
    - **git commit -m "Add LICENSE file"**
    - **git log --graph --oneline --decorate --all**

Name	Date modified	Type	Size
LICENSE	23/10/2018 1:39 AM	File	0 KB
myinfo	23/10/2018 12:18	Text Document	1 KB
README.md	22/10/2018 10:33	MD File	1 KB

Command Prompt

```
E:\Fall 2018\WAD\Assignment 1\testing git\test_git_l1f16bscs0003>git add LICENSE

E:\Fall 2018\WAD\Assignment 1\testing git\test_git_l1f16bscs0003>git commit -m "LICENSE file added"
[license 7e8a9f0] LICENSE file added
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 LICENSE

E:\Fall 2018\WAD\Assignment 1\testing git\test_git_l1f16bscs0003>git log --graph --oneline --decorate --all
* 7e8a9f0 (HEAD -> license) LICENSE file added
| * 6a4a3ea (testing) test.txt file added
|/
*   efe83a4 (origin/master, origin/HEAD, master) name conflict resolved
|\
| * 3e47fec name update
* | ab36430 Name is updated
|/
* 745d8f9 update info
* b8c7229 Update info
* b727146 My Personal Information
* 88f4554 My Personal Information
* 4816403 Initial commit

E:\Fall 2018\WAD\Assignment 1\testing git\test_git_l1f16bscs0003>
```

Command Prompt

```
E:\Fall 2018\WAD\Assignment 1\testing git\test_git_l1f16bscs0003>echo "License Version 1.0" >> LICENSE

E:\Fall 2018\WAD\Assignment 1\testing git\test_git_l1f16bscs0003>git add LICENSE

E:\Fall 2018\WAD\Assignment 1\testing git\test_git_l1f16bscs0003>git commit -m "license version added"
[license dea6289] license version added
 1 file changed, 1 insertion(+)

E:\Fall 2018\WAD\Assignment 1\testing git\test_git_l1f16bscs0003>git log --graph --oneline --decorate --all
* dea6289 (HEAD -> license) license version added
* 7e8a9f0 LICENSE file added
| * 6a4a3ea (testing) test.txt file added
|/
*   efe83a4 (origin/master, origin/HEAD, master) name conflict resolved
|\
| * 3e47fec name update
* | ab36430 Name is updated
|/
* 745d8f9 update info
* b8c7229 Update info
* b727146 My Personal Information
* 88f4554 My Personal Information
* 4816403 Initial commit

E:\Fall 2018\WAD\Assignment 1\testing git\test_git_l1f16bscs0003>
```

- Now switch back to our *master* branch. We do not have "test.txt" and "LICENSE" there. We can merge them to our *master* branch with the git merge command.
    - **git checkout master**
    - **git merge testing**
    - **git merge license**
    - **git log --graph --oneline --decorate --all**

```
Command Prompt

E:\Fall 2018\WAD\Assignment 1\testing git\test_git_l1f16bscs0003>git checkout master
Switched to branch 'master'
Your branch is up to date with 'origin/master'.

E:\Fall 2018\WAD\Assignment 1\testing git\test_git_l1f16bscs0003>git merge testing
Updating efe83a4..6a4a3ea
Fast-forward
 test.txt | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 test.txt

E:\Fall 2018\WAD\Assignment 1\testing git\test_git_l1f16bscs0003>git merge license
Merge made by the 'recursive' strategy.
 LICENSE | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 LICENSE

E:\Fall 2018\WAD\Assignment 1\testing git\test_git_l1f16bscs0003>git log --graph --oneline --decorate --all
*   381d57d (HEAD -> master) Merge branch 'license'
|\
| * dea6289 (license) license version added
| * 7e8a9f0 LICENSE file added
* | 6a4a3ea (testing) test.txt file added
|/
*   efe83a4 (origin/master, origin/HEAD) name conflict resolved
|\
| * 3e47fec name update
* | ab36430 Name is updated
|/
* 745d8f9 update info
* b8c7229 Update info
* b727146 My Personal Information
* 88f4554 My Personal Information
* 4816403 Initial commit

E:\Fall 2018\WAD\Assignment 1\testing git\test_git_l1f16bscs0003>
```

- Now, you can delete testing and license branch with **git branch -d <branch>**
    - **git branch -d testing**
    - **git branch -d license**
    - **git log --graph --oneline --decorate --all**

```
Command Prompt

E:\Fall 2018\WAD\Assignment 1\testing git\test_git_l1f16bscs0003>git branch -d testing
Deleted branch testing (was 6a4a3ea).

E:\Fall 2018\WAD\Assignment 1\testing git\test_git_l1f16bscs0003>git branch -d license
Deleted branch license (was dea6289).

E:\Fall 2018\WAD\Assignment 1\testing git\test_git_l1f16bscs0003>git log --graph --oneline --decorate --all
*   381d57d (HEAD -> master) Merge branch 'license'
|\
| * dea6289 license version added
| * 7e8a9f0 LICENSE file added
* | 6a4a3ea test.txt file added
|/
*   efe83a4 (origin/master, origin/HEAD) name conflict resolved
|\
| * 3e47fec name update
* | ab36430 Name is updated
|/
* 745d8f9 update info
* b8c7229 Update info
* b727146 My Personal Information
* 88f4554 My Personal Information
* 4816403 Initial commit

E:\Fall 2018\WAD\Assignment 1\testing git\test_git_l1f16bscs0003>
```

9. Assignment Submission

You have to submit this assignment in the two following ways.

I. Upload a ZIP file to your Github repository

The simplest, safest way is to export your project to a zip file and upload it (as the "myinfo.txt" file). In this example, I exported my screenshots to "*name_reg*_Assignment1.zip" and copied it to test_git_*registration_number* folder.

```
Command Prompt

E:\Fall 2018\WAD\Assignment 1\testing git\test_git_l1f16bscs0003>git status
On branch master
Your branch is up to date with 'origin/master'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        MuhammadAliMakhdoom_L1F16BSCS0003_Assignment 1.zip

nothing added to commit but untracked files present (use "git add" to track)

E:\Fall 2018\WAD\Assignment 1\testing git\test_git_l1f16bscs0003>git add "MuhammadAliMakhdoom_L1F16BSCS0003_Assignment 1.zip"

E:\Fall 2018\WAD\Assignment 1\testing git\test_git_l1f16bscs0003>git commit -m "Assignment 1 zip file added"
[master a726542] Assignment 1 zip file added
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 MuhammadAliMakhdoom_L1F16BSCS0003_Assignment 1.zip

E:\Fall 2018\WAD\Assignment 1\testing git\test_git_l1f16bscs0003>git push
```

- Now, add the file, commit and push the change to your remote repository.
  - **git add MuhammadAliMakhdoom_L1F16BSCS0003_Assignment1.zip**
  - **git commit -m "Assignment 1 zip file added"**

- **git push**

II. Submit Zip file to LMS before the deadline

You have to copy the entire local repository folder, all screenshots, and explanation in word file also having URL of your remote repository on Github.
NOTE: Make sure that you added everything and that it works as expected.

10. Further Readings

- https://git-scm.com/book/en/v2
- https://services.github.com/on-demand/downloads/github-git-cheat-sheet.pdf

HAVE FUN ;-)