**CS421/621**

**Lab 01 – Introduction to Git & GitLab**

**Objectives**

* Getting Started with the Git -version control tool-.
* GitLab
* Github

**Using software versioning tool- git**

Git is a popular software version control system that allows collaboration among large group of software developers. You can use git to keep track of your own software as well as share and collaborate with other developers. It also allows you to keep a track of changes made to a project over time. It works by recording the changes you make to a project, storing those changes, then allowing you to reference them as needed. Now, let’s start using git on the command line.

**How to install git**

Mac & Linux should have git installed already, to verify this run *$git* or *$git –version* command in terminal.

**Installation on Linux:**

For Debian/Ubuntu use: *$apt-get install git*

For Redhat/Centos use: *$yum install git*

For any other flavor refer to<https://git-scm.com/download/linux>

**Installation on Mac:** From Terminal run *$git –version,* this will prompt for installation if not already available.

**Installation on Windows:** Download build available on the Git website: <https://git-scm.com/download/win>

**\*More info about installation:** <https://git-scm.com/book/en/v2/Getting-Started-Installing-Git>

**Using software versioning tool- git**

Git is a popular software version control system that allows collaboration among large group of software developers. You can use git to keep track of your own software as well as share and collaborate with other developers. It also allows you to keep a track of changes made to a project over time. It works by recording the changes you make to a project, storing those changes, then allowing you to reference them as needed. Now, let’s start using git on the command line.

To use git on the command line, check git version, add your git username and set your email and perform other important things, please click on the link provided below and follow the steps and instructions given and perform the stuffs.

<https://docs.gitlab.com/ee/gitlab-basics/start-using-git.html>

It’s quite easy, right?? Now, lets go deep.

A Git project can be thought of as having three parts:

1. A Working Directory: where you’ll be doing all the work: creating, editing, deleting and organizing files.

2. A Staging Area: where you’ll list changes, you make to the working directory.

3. A Repository: where Git permanently stores those changes as different versions of the project.

The Git workflow consists of editing files in the working directory, adding files to the staging area, and saving changes to a Git repository.

Our next step should be executing the basic commands of git and knowing them. Below are few commands we are going to use in this semester and going forward.

**1. git config:** One of the most used git commands is git config which can be used to set user-specific configuration values like email, preferred algorithm for diff, username and file format etc. For example, the following command can be used to set the user name and email:

git config --global user.name "mahmut\_unan"

git config --global user.email "unan@uab.edu"

**2. git init:** This command is used to create a new GIT repository. The word "init" means initialize. The command sets up all the tools Git needs to begin tracking changes made to the project.

**3. git add:** The git add command can be used in order to add files to the index. For example, the following command will add a file named temp.txt present in the local directory to the index: git add temp.txt

**4. git clone:** The git clone command is used for repository checking out purposes. If the repository lies on a remote server, use: git clone alex@93.188.160.58:/path/to/repository (this is just an example, you must copy paste the path which you want to clone with git). Conversely, if a working copy of a local repository is to be created, use: git clone /path/to/repository

**5. git commit:** The git commit command is used to commit the changes to the head. Note that any committed changes won’t make their way to the remote repository. Usage: git commit –m “Message to go with the commit here”

**6. git status:** The git status command displays the list of changed files along with the files that are yet to be added or committed. Usage: git status

**7. git push:** git push is another one of the most used basic git commands. A simple push sends the made changes to the master branch of the remote repository associated with the working directory. For example: git push origin master

**8. git checkout:** The git checkout command can be used to create branches or to switch between them. For example, the following creates a new branch and switches to it: command git checkout -b <branch-name>. To simply switch from one branch to another use: git checkout <branch-name>

**9. git remote:** The git remote command lets a user connect to a remote repository. The following command lists the remote repositories that are currently configured: git remote –v This command allows the user to connect the local repository to a remote server: git remote add origin <93.188.160.58>

**10. git branch:** The git branch command can be used to list, create or delete branches. To list all the branches, present in the repository use: git branch

To delete a branch: git branch –d <branch-name>

**11. git pull:** In order to merge all the changes, present on the remote repository to the local working directory, the pull command is used. Usage: git pull

**12. git merge:** The git merge command is used to merge a branch into the active branch. Usage: git merge <branch-name>

**13. git diff:** The git diff command is used to list down conflicts. In order to view conflicts with the base file, use: git diff --base <file-name>

The following command is used to view the conflicts between about-to-be-merged branches prior to merging them:

git diff <source-branch> <target-branch>

To simply list down all the present conflicts, use: git diff

**14. git tag:** Tagging is used to mark specific commits with simple handles. An example can be: git tag 1.1.0 <insert-commitID-here>

**15. git log:** Running the git log command outputs a list of commits on a branch along with pertinent details. A sample output can be: commit 15f4b6c44b3c8344caasdac9e4be13246e21sadw

Author: Alex Hunter <alexh@gmail.com>

Date: Mon Oct 1 12:56:29 2016 -0600

**17. git reset:** To reset the index and the working directory to the last commit’s state, git reset command is used. Usage: git reset --hard HEAD

**18. git rm:** git rm can be used to remove files from the index and the working directory. Usage: git rm filename.txt

**19. git stash:** Probably one of the lesser-known basic git commands is git stash which helps in saving changes that are not to be committed immediately, but on a temporary basis. Usage: git stash

**20. git show:** In order to view information about any git object, use the git show command. For example: git show

**21. git fetch:** git fetch allows a user to fetch all those objects from the remote repository that doesn’t currently reside in the local working directory. Example usage: git fetch origin

**22. git cat-file:** Using the SHA-1 value, view the type of an object by using the git cat-file command. For example: git cat-file –p d670460b4b4aece5915caf5c68d12f560a9fe3e4

**23. git grep:** git grep lets a user search through the content trees for phrases and/or words. For example, to search for www.hostinger.com in all files use: git grep "www.hostinger.com"

**24. gitk:** gitk is the graphical interface for a local repository that can be invoked by typing and running: gitk

**25. git instaweb:** With the git instaweb command, a web server can be run interfaced with the local repository. A web browser is also automatically directed to it. For instance: git instaweb –httpd=webrick

**26. git gc:** To optimize the repository via garbage collection, which will clean up unneeded files and optimize them, use: git gc

**27. git archive:** The git archive command lets a user create a zip or a tar file containing the constituents of a single repository tree. For instance: git archive --format=tar master

**28. git rebase:** The git rebase command is used for reapplication of commits on another branch. For instance: git rebase master

**29. git prune:** Via the git prune command, objects that don’t have any incoming pointers are deleted. Usage: git prune

**30. git fsck:** In order to perform an integrity check of the git file system, use the command git fsck. Any corrupted objects are identified: git fsck

Apart from these, if you want to learn more about git, please make sure you check below tutorials: <https://www.codecademy.com/learn/learn-git> and <https://www.atlassian.com/git>. If you did not use Git before, read the Git book: <https://git-scm.com/book/en/v2> . Minimally, Chapter 1 and Chapter 2 are required for this class.

**How to create an account in GitLab?**

GitLab is a self-hosted Git-repository management system that keeps the user code private and can easily deploy the changes of the code. Our department provides a version of GitLab (UAB CS GitLab). Click on the following link <https://gitlab.cis.uab.edu/> and you will use your BlazerID to log in. For a new project, click **new project** and name it **CS421 or CS621.** Then, click **create project.**

**Configuration Part**

1. Open terminal (Linux/Mac) or git bash (Windows)
2. Type the following commands:

***$git config --global user.name*** *yourName*

***$git config --global user.email***[*yourBlazerID@uab.edu*](mailto:yourBlazerID@uab.edu)

1. Don’t copy and paste, replace yourName with your name and yourBlazerID with your BlazerID.
2. Type ***$ssh-keygen -t rsa -C*** *yourBlazerID@uab.edu -b 4096*

// NOTE: There is NO space between ssh and -keygen//

1. Enter ***SSHKey*** for the file name that your key in it. (Now, your key in ***SSHKey.pub*** file)
2. Type ***$cat ~/.ssh/id\_rsa.pub***. You shall see a string starting with ***ssh-rsa*** and ending with **your email**. Copy the string starting with *ssh-rsa* and ending with your email. *(if you can’t open this file from gitbash, find the file and open it with a text editor)*
3. Go back to <https://gitlab.cis.uab.edu/>
4. At the upper right corner, click the icon 
5. Click **Settings**.
6. Click **SSH Keys** on the left side.
7. Paste the key in the box. Click **add key.**

**\*More info about SSH Key:** <https://docs.gitlab.com/ee/ssh/>

**Use git**

1. In your browser, go to <https://gitlab.cs.uab.edu/CS421-621/Lab01>
2. Click Fork. Now you created a fork of the project.
3. Click your user name.
4. You shall see a link like this: git@gitlab.cs.uab.edu:[*yourBlazerID*](mailto:yourBlazerID@uab.edu)/Lab01.git in the middle of the page. Copy this link.
5. Return to your terminal or git bash.
6. Input git clone git@gitlab.cs.uab.edu:[*yourBlazerID*](mailto:yourBlazerID@uab.edu)/Lab01.git
7. After pressing enter, you shall see the project being copied to your local machine.

**Grant permission:** [Note: This step is necessary for all future homework and projects. Otherwise, your submission cannot be graded.]

1. In your browser, go to the project your just forked.
2. Click Setting ->> Permissions ->> Project visibility = set to private
3. Click Settings ->> Members.
4. In the search bar, input Unan. Click Mahmut Unan. (Similarly, add Anuradha Mandal anuradha@uab.edu and Trupeshkumar Rajendrabhai Patel tr27p@uab.edu)
5. At the next drop-down menu, select Reporter.
6. Click Add to project.

**Exercise**

Clone lab1 project folder and modify the index.html file. Change the title with your name. Once you finish your task, commit/push the changes.

**GitHub (Not required for the lab)**

Creating an account on GitHub:

Github accounts are free for everyone. You may have any number of public or private repositories, but you are limited with 3-collaborators for the private repositories (you need to pay fees to have more collaborators, see <https://github.blog/2019-01-07-new-year-new-github/>).

Please follow the instructions at the following page to create a free account:

<https://help.github.com/en/articles/signing-up-for-a-new-github-account>

Here is the link to create a free account

<https://github.com/join>

Suggested links to learn more:

<https://try.github.io/>

<https://classroom.udacity.com/courses/ud775>

\*\*\* Feel free to use Canvas Discussion board for the questions/recommendations..etc\*\*\*