

CS421/621

Lab 01 – Introduction to Git & GitHub

Objectives

- Getting Started with the Git -version control tool-.
- 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.

Github

Creating an account on GitHub:

Use education package (free pro account)

<https://education.github.com/pack>

Lab 1 - 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 setup for Lab Assignments :

1. Open terminal (Linux/Mac) or git bash (Windows)
2. Type the following commands and press enter.
Git clone <https://github.com/pranith1996/lab1>

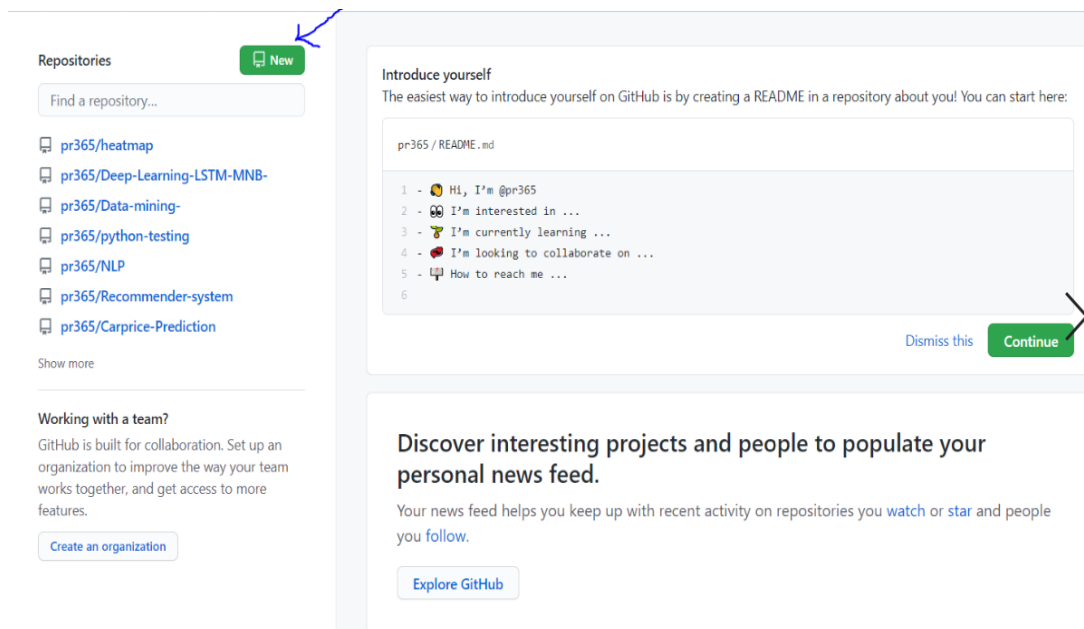
```
UAB+pranith@SLP-20990544 MINGW64 ~  
$ git clone https://github.com/pranith1996/lab1
```

Once you complete doing above commands you should get the lab1 folder to your local system

3. Type the following command to remove the origin
`git remote rm origin`

```
UAB+pranith@SLP-20990544 MINGW64 ~  
$ cd lab1  
  
UAB+pranith@SLP-20990544 MINGW64 ~/lab1 (main)  
$ git remote rm origin
```


4. Go to your github account and create a private repository (see the screenshots to create private repository).



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner *

 pr365

Repository name *

/ lab1 

Great repository names are short and memorable. Need inspiration? How about [fluffy-couscous?](#)

Description (optional)



Public

Anyone on the internet can see this repository. You choose who can commit.



Private

You choose who can see and commit to this repository.

Initialize this repository with:

Skip this step if you're importing an existing repository.

☐ Add a README file

This is where you can write a long description for your project. [Learn more.](#)

☐ Add .gitignore


Choose which files not to track from a list of templates. [Learn more.](#)

☐ Choose a license

A license tells others what they can and can't do with your code. [Learn more.](#)



Create repository

5. Copy your private repository url which you have created in the step4 (highlighted in yellow is the url for your repository)

 pr365 / lab1 PrivateUnwatch 1Star 0

[Code](#) [Issues](#) [Pull requests](#) [Actions](#) [Projects](#) [Security](#) [Insights](#) [Settings](#)


Quick setup — if you've done this kind of thing before

 Set up in Desktop or [HTTPS](#) [SSH](#) <https://github.com/pr365/lab1.git> 

Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

...or create a new repository on the command line

```
echo "# lab1" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/pr365/lab1.git
git push -u origin main
```



6. Make the necessary code changes in index file
7. Type the following commands (Adding origin to push the code to your private repository)

`git remote add origin yoururl(copied url from step5)`

```
UAB+pranith@SLP-20990544 MINGW64 ~/lab1 (main)
$ git remote add origin https://github.com/pr365/lab1.git
```

8. Type the following commands (Before typing the below commands make sure your current directory is lab1 folder in local machine).

```
git add *
git commit -m 'message'
git push --set-upstream origin main
```

```
UAB+pranith@SLP-20990544 MINGW64 ~/lab1 (main)
$ git add *

UAB+pranith@SLP-20990544 MINGW64 ~/lab1 (main)
$ git commit -m 'message'
[main a7cb9c6] message
1 file changed, 2 insertions(+), 2 deletions(-)

UAB+pranith@SLP-20990544 MINGW64 ~/lab1 (main)
$ git push --set-upstream origin main
```

9. Now go to settings →manage access →click on invite collaborator and add below users
 - 1.pranith1996
 - 2.mahmut-unan
 - 3.zaid-ali

pr365 / lab1 Private

<> Code ⓘ Issues 🔗 Pull requests ▶ Actions 📁 Projects ⓘ Security 📈 Insights ⚙️ Settings

main 1 branch 0 tags

Go to file Add file Code

pranith1996 message a7cb9c6 12 minutes ago 2 commits

index.html	message	12 minutes ago
lab01.docx	Add files via upload	16 hours ago

Add a README with an overview of your project. Add a README

pr365 / lab1 Private

<> Code ⓘ Issues 🔗 Pull requests ▶ Actions 📁 Projects ⓘ Security 📈 Insights ⚙️ Settings

Options

Manage access

Security & analysis

Branches

Webhooks

Notifications

Integrations

Deploy keys

Actions

Settings

Repository name

lab1 Rename

☐ Template repository
Template repositories let users generate new repositories with the same directory structure and files. [Learn more.](#)

Social preview

⚠️ You can upload a social image, but it will not be visible publicly while pr365/lab1 is private.

Upload an image to customize your repository's social media preview.
Images should be at least 640×320px (1280×640px for best display).
[Download template](#)

Options

Manage access

Security & analysis

Branches

Webhooks

Notifications

Integrations

Deploy keys

Actions

Secrets

Pages

Who has access

PRIVATE REPOSITORY

Only those with access to this repository can view it.

Manage

DIRECT ACCESS

0 collaborators have access to this repository. Only you can contribute to this repository.

Manage access

You haven't invited any collaborators yet

Invite a collaborator

After clicking invite a collaborator the below tab pops up and enter the username of Professor and TA's and add to your repository.

×

Invite a collaborator to lab1

pranith1996

pranith1996

Invite collaborator

11.Finally submit the link(url of your repository)

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***