

LAB - 302-2.1 - Hands-on Activity Git and GitHub

Created by Muhammad Haseeb

Local repository

1) Create a local git repository:

When creating a new project on your local machine using git, you'll first create a new repository (or often, 'repo', for short).

To initialize a git repository in the root of the folder, run the git init command.

git init

2) Create a new file in the repo

Create a new file for the project using any text editor. Let us just create and save a blank file named newfile.txt.

Once you've created or modified files in a folder containing a git repo, git will notice that the file exists inside the repo. But git won't track the file unless you explicitly tell it to. Git only saves/manages changes to files that it tracks, so we'll need to send a command to confirm that, yes, we want git to track our new file.

3) Add files to the staging area.

To add a file to a commit, you must first add it to the staging environment. To do this, you can use the **git add <filename>** command

git add newfile.txt

After creating the new file, you can use the git status command to see which files git knows to exist.

git status

or

git log --oneline

```
C:\Users>cd PSAdmin
C:\Users\PSAdmin>cd githubdemo
C:\Users\PSAdmin\githubdemo>git init
Initialized empty Git repository in C:/Users/PSAdmin/githubdemo/.git/
C:\Users\PSAdmin\githubdemo>ls
C:\Users\PSAdmin\githubdemo>git add newfiles.txt
fatal: pathspec 'newfiles.txt' did not match any files
C:\Users\PSAdmin\githubdemo>git add newfile.txt
C:\Users\PSAdmin\githubdemo>git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

        new file:   newfile.txt

C:\Users\PSAdmin\githubdemo>
```

There are several commands that you can choose from. As always, it's important to know what you are staging and committing.

“git add -A” stages all files, including new, modified, and deleted files, including files in the current directory and in higher directories that still belong to the same git repository.

"git add ." adds the entire directory recursively, including files whose names begin with a dot(.).

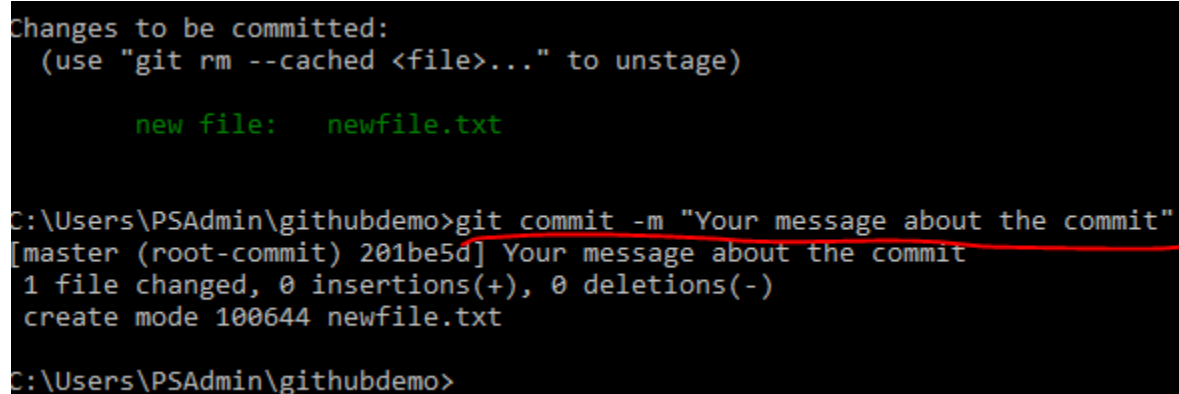
"git add -u" stages new and modified files only, NOT deleted files.

For more: <https://github.com/git-guides/git-add>

4) Create a commit

It's time to create your first commit! Run the below command

git commit -m "Your message about the commit"

A terminal window with a black background and white text. It shows the output of a git commit command. The first line is "Changes to be committed:" followed by "(use 'git rm --cached <file>...' to unstage)". Below that, it says "new file: newfile.txt". Then, the command "C:\Users\PSAdmin\githubdemo>git commit -m 'Your message about the commit'" is entered. The output shows "[master (root-commit) 201be5d] Your message about the commit", "1 file changed, 0 insertions(+), 0 deletions(-)", and "create mode 100644 newfile.txt". The prompt "C:\Users\PSAdmin\githubdemo>" is shown at the bottom.

```
Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

       new file:   newfile.txt

C:\Users\PSAdmin\githubdemo>git commit -m "Your message about the commit"
[master (root-commit) 201be5d] Your message about the commit
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 newfile.txt

C:\Users\PSAdmin\githubdemo>
```

Run the below command to see a **list of all the commits made to a repository.**

git log

5) Make changes in newfile.txt

Let's add new content or some lines in the newfile.txt, then try to commit new content to the git repository.

git commit -m "Your 2nd message about the commit"

You will get an error if you run the above command because you have to put your file on the staging area first. Then you will be able to run the commit command. To do so, run the below commands.

```
git add newfile.txt
```

```
git commit -m "Your 2nd message about the commit"
```

Or

Git commit -a -m "Your 2nd message about the commit" → This option allows you to skip the staging phase. The addition of -a will automatically stage any files that are already being tracked by Git (changes to files that you've committed before).

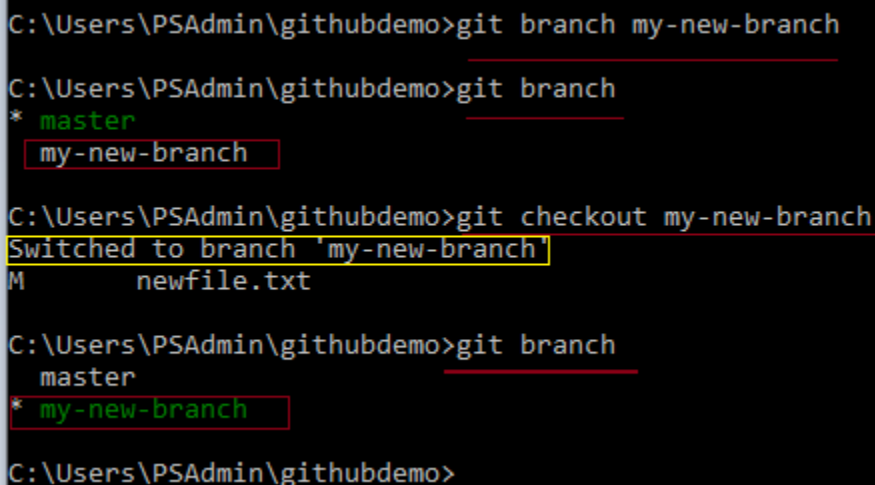
6) Create a new branch and switch to a new branch

```
git branch my-new-branch // create a new branch
```

```
git branch // show all list of branches
```

```
git checkout my-new-branch // switch master branch to new branch
```

```
git branch // show all list of branches
```



```
C:\Users\PSAdmin\githubdemo>git branch my-new-branch
C:\Users\PSAdmin\githubdemo>git branch
* master
  my-new-branch
C:\Users\PSAdmin\githubdemo>git checkout my-new-branch
Switched to branch 'my-new-branch'
M       newfile.txt
C:\Users\PSAdmin\githubdemo>git branch
  master
* my-new-branch
C:\Users\PSAdmin\githubdemo>
```

7) Make changes in newfile.txt

Let's add new content or some lines in the newfile.txt, then try to commit new content on the git repository, but this time, all changes will be saved on the new branch.

git add newfile.txt

git commit -m "add changes in txt file and commit in new branch"

8) Switch back to the master branch.

git checkout main

9) Merging branches

git merge <branch Name> command is responsible for merging two branch

git merge my-new-branch

10) Delete newly created branch.

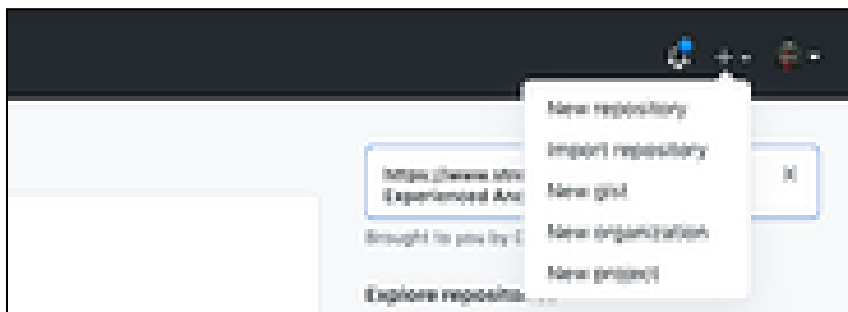
git branch -d my-new-branch

Remote Repository

1. Create a new repository on GitHub

If you only want to keep track of your code locally, you don't need to use GitHub. But if you want to work with a team, you can use GitHub to collaboratively modify the project's code.

To create a new repo on GitHub, log in and go to the GitHub home page. You can find the **"New repository"** option under the **"+"** sign next to your profile picture, in the top right corner of the navbar:




After clicking the button, GitHub will ask you to name your repo and provide a brief description:


Create a new repository

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

Owner *

Repository name *


 dfryer1193 ▾

/ newrepo 


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

Description (optional)

My new repo!

☒  **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

When you're done filling out the information, press the 'Create repository' button to create your new repository.

2. Push a branch to GitHub

```
git remote add origin https://github.com/Username/demoforexercise.git
```

```
git push -u origin main
```

Username:

password

```
git remote add origin  
https://github.com/UserAccountName/demoforexcercise.git -
```

The above command tells our local repository about a remote repository located somewhere. The location of our remote repository is `https://github.com/UserAccountName/demoforexcercise.git`. Now if we want to send our code to GitHub we can just type in `git push` `https://github.com/UserAccountName/demoforexcercise.git main` - but typing that whole URL is quite a pain. Instead, it would be better if we could give the URL a nickname (also called an alias) - that is what "origin" is! By setting up this alias, we can simply type `git push origin main` to send the code from our `main` branch to this remote repository.

So going back, `git remote add` is how we tell our local repository about a remote repository (that we can send/retrieve code from). `origin` is a nickname for where the repository is actually located (at

```
https://github.com/UserAccountName/demoforexcercise.git).
```

To see your remotes locally, you can type `git remote -v`. If you need to remove a remote, you can use `git remote rm NAME_OF_REMOTE`

```
git push -u origin main
```

 - Now we can send our code from a local repository to a remote repository (which we aliased to `origin` in the previous command). The `-u` flag allows us in the future to only have to type `git push` instead of `git push origin main`.

So to review this command - `git push` is how we send code from a local repository to a remote repository. `origin` is where we are sending it (specifically to `origin/main` which is the master branch of our remote repository). `main` is the local branch which we are sending our code from.

3. Get changes on GitHub back to your computer.

```
git pull origin main
```

Or

```
git pull
```