



# Guided LAB - 302.1.2 - Hands-on Activity Git and GitHub

---

## Introduction:

While Git is a tool that's used to manage multiple versions of source code edits that are then transferred to files in a Git repository, GitHub serves as a location for uploading copies of a Git repository.

## Lab Objective:

In this lab, we will demonstrate how to create a local repository for the project using Git, and how to push the project on github from the local repository.

## Learning Objective:

By the end of this lab, learners will be able to use Git and GitHub with the Project.

## Instructions:

### Part 1: Local repository

#### 1. Create a local Git repository

When creating a new project on your local machine using Git, you will 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's just create and save a blank file named newfile.txt.

Once you have created or modified files in a folder containing a git repo, git will notice that the file exists inside the repo. But git will not track the file unless you explicitly tell it



to. Git only saves/manages changes to files that it tracks, so we will 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 area/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 are in existence.

***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 is important to know what you are staging and committing.

“**git add -A**” stages all files, including new, modified, and deleted files, as well as 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; it does NOT stage deleted files.

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

## 4. Create a commit

It is time to create your first commit! Run the command below:

***git commit -m "Your message about the commit"***

```
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 command below to see a list of all of 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, and 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 have committed before).

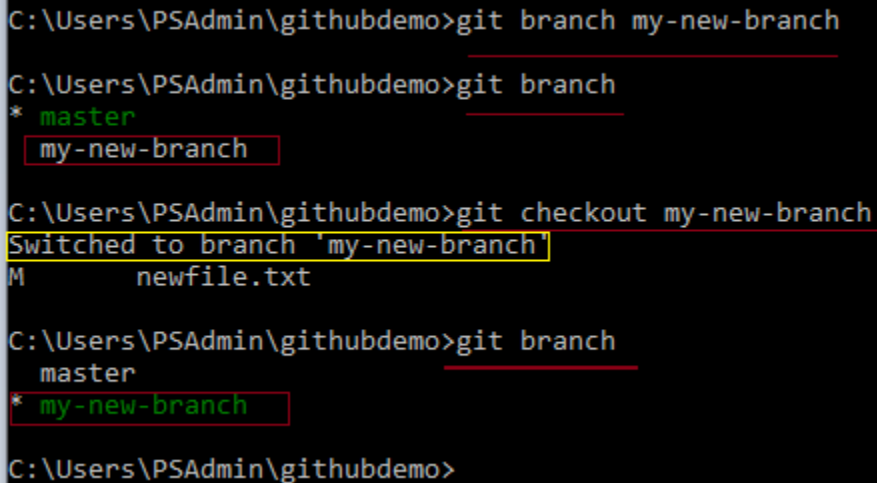
## 6. Create a new branch and switch to a new branch

```
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, and then try to commit new content on the git repository; however, 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.

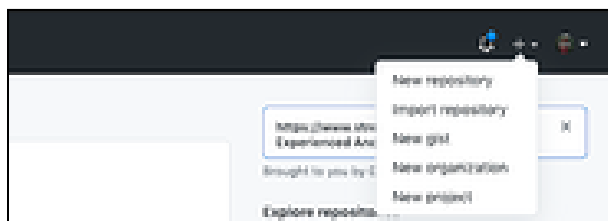
*git branch -d my-new-branch*

# Part 2: Remote Repository

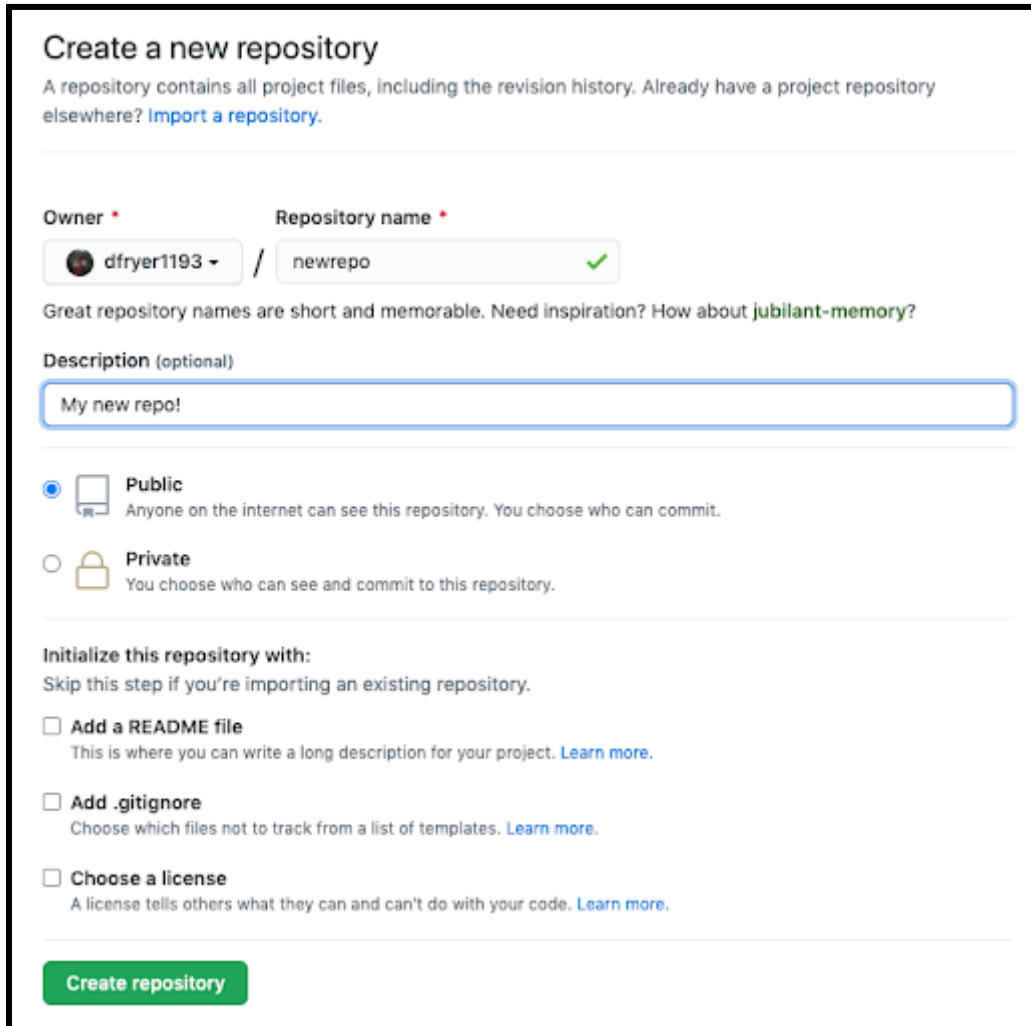
## 1. Create a new repository on GitHub

If you only want to keep track of your code locally, you do not 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 navigation bar:



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



The screenshot shows the GitHub 'Create a new repository' page. At the top, it says 'Create a new repository' and provides a brief explanation of what a repository is. Below this, there are two main input fields: 'Owner' and 'Repository name'. The 'Owner' field is set to 'dfryer1193' and the 'Repository name' field is set to 'newrepo', which has a green checkmark next to it. Below these fields, there is a suggestion for repository names: 'Great repository names are short and memorable. Need inspiration? How about [jubilant-memory](#)?'. The 'Description (optional)' field contains the text 'My new repo!'. Below the description field, there are two radio button options for repository visibility: 'Public' (selected) and 'Private'. The 'Public' option is described as 'Anyone on the internet can see this repository. You choose who can commit.' The 'Private' option is described as 'You choose who can see and commit to this repository.' Below these options, there is a section titled 'Initialize this repository with:' which includes three checkboxes: 'Add a README file' (unchecked), 'Add .gitignore' (unchecked), and 'Choose a license' (unchecked). Each checkbox has a brief description and a 'Learn more' link. At the bottom of the form is a green button labeled 'Create repository'.

When you finish 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/demoforexercise.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/demoforexercise.git`. Now, if we want to send our code to GitHub, we can just type in `git push https://github.com/UserAccountName/demoforexercise.git main` - but typing that whole URL can be difficult. 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 the 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/demoforexercise.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`.

Let's 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 from which we are sending our code.

### 3. Get changes on GitHub back to your computer.

```
git pull origin main
```

Or

```
git pull
```



## Submission Requirements:

Prior to beginning this guided lab, download a copy for yourself and save to your desktop/laptop, Google Drive, or Microsoft OneDrive. Throughout the lab, you will be required to take screenshots of your progress, and insert the screenshots in the designated areas.

After completing the guided lab, submit your copy of the document on Canvas. You will be prompted to complete a file upload. By uploading the document with the required screenshots, the guided lab can be reviewed by the instructor and marked as completed.

Take a screenshot of the report appearing in the test results, and enter the image below.



```

ungraded-java-sba
Johns-MacBook-Pro:Desktop johnppkyaw$ cd Per\ Scholas/
Johns-MacBook-Pro:Per Scholas johnppkyaw$ mkdir 302
Johns-MacBook-Pro:Per Scholas johnppkyaw$ cd 302
Johns-MacBook-Pro:302 johnppkyaw$ mkdir VersionControlExercise && cd VersionControlExercise
Johns-MacBook-Pro:VersionControlExercise johnppkyaw$ git init
Initialized empty Git repository in /Users/johnppkyaw/Desktop/Per Scholas/302/VersionControlExercise/.git/
Johns-MacBook-Pro:VersionControlExercise johnppkyaw$ touch newfile.txt
Johns-MacBook-Pro:VersionControlExercise johnppkyaw$ git add newfile.txt
Johns-MacBook-Pro:VersionControlExercise johnppkyaw$ git status
On branch main

No commits yet

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

Johns-MacBook-Pro:VersionControlExercise johnppkyaw$ git commit -m "first commit"
[main (root-commit) 1eb95ad] first commit
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 newfile.txt
Johns-MacBook-Pro:VersionControlExercise johnppkyaw$ git log
commit 1eb95add97a5faaee83798a0c0883366c489d4ba (HEAD -> main)
Author: John <jjpyae@gmail.com>
Date:   Fri Dec 8 12:23:50 2023 -0800

    first commit
Johns-MacBook-Pro:VersionControlExercise johnppkyaw$ code .
Johns-MacBook-Pro:VersionControlExercise johnppkyaw$ git commit -m "2nd commit"
On branch main
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
    modified:   newfile.txt

no changes added to commit (use "git add" and/or "git commit -a")
Johns-MacBook-Pro:VersionControlExercise johnppkyaw$ git add .
Johns-MacBook-Pro:VersionControlExercise johnppkyaw$ git commit -m "second commit"
[main 442a9fc] second commit
 1 file changed, 1 insertion(+)
Johns-MacBook-Pro:VersionControlExercise johnppkyaw$ git log
commit 442a9fc3b1c0e5393ef0ab7f322fe61d1db8b804 (HEAD -> main)
Author: John <jjpyae@gmail.com>
Date:   Fri Dec 8 12:24:50 2023 -0800

    second commit

commit 1eb95add97a5faaee83798a0c0883366c489d4ba
Author: John <jjpyae@gmail.com>
Date:   Fri Dec 8 12:23:50 2023 -0800

    first commit
Johns-MacBook-Pro:VersionControlExercise johnppkyaw$ git branch mynewbranch
Johns-MacBook-Pro:VersionControlExercise johnppkyaw$ git branch
* main
  mynewbranch
Johns-MacBook-Pro:VersionControlExercise johnppkyaw$ git checkout mynewbranch
Switched to branch 'mynewbranch'
Johns-MacBook-Pro:VersionControlExercise johnppkyaw$ git branch
  main
* mynewbranch
Johns-MacBook-Pro:VersionControlExercise johnppkyaw$ git add .
Johns-MacBook-Pro:VersionControlExercise johnppkyaw$ git commit -m "edit made in mynewbranch"
[mynewbranch d46fa75] edit made in mynewbranch
 1 file changed, 1 insertion(+)
Johns-MacBook-Pro:VersionControlExercise johnppkyaw$ git checkout main
Switched to branch 'main'
Johns-MacBook-Pro:VersionControlExercise johnppkyaw$ git merge mynewbranch
Updating 442a9fc..d46fa75
Fast-forward
 newfile.txt | 1 +
 1 file changed, 1 insertion(+)
Johns-MacBook-Pro:VersionControlExercise johnppkyaw$ git branch -d mynewbranch
Deleted branch mynewbranch (was d46fa75).
Johns-MacBook-Pro:VersionControlExercise johnppkyaw$ git remote add origin git@github.com:johnppkyaw/VersionControlExercise.git
Johns-MacBook-Pro:VersionControlExercise johnppkyaw$ git push -u origin main
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (9/9), 669 bytes | 669.00 KiB/s, done.
Total 9 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:johnppkyaw/VersionControlExercise.git
 * [new branch] main -> main
branch 'main' set up to track 'origin/main'.
Johns-MacBook-Pro:VersionControlExercise johnppkyaw$

```