Experiment 03 - Git Commands

Roll No.	24
Name	Iyer Sreekesh Subramanian
Class	D15-A
Subject	DevOps Lab
LO Mapped	LO1: To understand the fundamentals of DevOps engineering and be fully proficient with DevOps terminologies, concepts, benefits, and deployment options to meet your business requirements LO2: To obtain complete knowledge of the “version control system” to effectively track changes augmented with Git and GitHub

Aim:

To Perform various GIT operations on local and Remote repositories using GIT Cheat-Sheet

Introduction:

Git commands are used like utilities to interact with our version control system
After installing git, one can get the list of all available commands by typing '*git --help*' in the CLI.

Following is the list of frequently used commands:

1. ***git add***
    a. Moves changes from the working directory to the staging area.
    b. This gives you the opportunity to prepare a snapshot before committing it to the official history.
2. ***git branch***
    a. This command is your general-purpose branch administration tool.
    b. It lets you create isolated development environments within a single repository.
3. ***git checkout***
    a. In addition to checking out old commits and old file revisions, git checkout is also the means to navigate existing branches.

   b. Combined with the basic Git commands, it's a way to work on a particular line of development.

4. ***git clean***
   a. Removes untracked files from the working directory.
   b. This is the logical counterpart to git reset, which (typically) only operates on tracked files.
5. ***git clone***
   Creates a copy of an existing Git repository.
   a. Cloning is the most common way for developers to obtain a working copy of a central repository.
6. ***git commit***
   Takes the staged snapshot and commits it to the project history.
   a. Combined with git add, this defines the basic workflow for all Git users.
7. ***git commit --amend***
   a. Passing the --amend flag to git commit lets you amend the most recent commit.
   b. This is very useful when you forget to stage a file or omit important information from the commit message.
8. ***git config***
   a. A convenient way to set configuration options for your Git installation.
   b. You'll typically only need to use this immediately after installing Git on a new development machine.
9. ***git fetch***
   a. Fetching downloads a branch from another repository, along with all of its associated commits and files. But, it doesn't try to integrate anything into your local repository.
   b. This gives you a chance to inspect changes before merging them with your project.

10. ***git init***
    a. Initializes a new Git repository. If you want to place a project under revision control, this is the first command you need to learn.
11. ***git log***
    a. Lets you explore the previous revisions of a project.
    b. It provides several formatting options for displaying committed snapshots.

12. ***git merge***
    a. A powerful way to integrate changes from divergent branches.
    b. After forking the project history with the git branch, git merge lets you put it back together again.
13. ***git pull***

a. Pulling is the automated version of git fetch.
b. It downloads a branch from a remote repository, then immediately merges it into the current branch.
c. This is the Git equivalent of svn update.

14. ***git push***

a. Pushing is the opposite of fetching (with a few caveats).
b. It lets you move a local branch to another repository, which serves as a convenient way to publish contributions.
c. This is like svn commit, but it sends a series of commits instead of a single changeset.

15. ***git rebase***

a. Rebasing lets you move branches around, which helps you avoid unnecessary merge commits.
b. The resulting linear history is often much easier to understand and explore.

16. ***git rebase -i***

a. The -i flag is used to begin an interactive rebasing session.
b. This provides all the benefits of a normal rebase, but gives you the opportunity to add, edit, or delete commits along the way.

17. ***git reflog***

a. Git keeps track of updates to the tip of branches using a mechanism called reflog.
b. This allows you to go back to changesets even though they are not referenced by any branch or tag.

18. ***git remote***

a. A convenient tool for administering remote connections.
b. Instead of passing the full URL to the fetch, pull, and push commands, it lets you use a more meaningful shortcut.

19. ***git reset***

a. Undoes changes to files in the working directory.
b. Resetting lets you clean up or completely remove changes that have not been pushed to a public repository.

20. ***git revert***

a. Undoes a committed snapshot.
b. When you discover a faulty commit, reverting is a safe and easy way to completely remove it from the codebase.

21. ***git status***

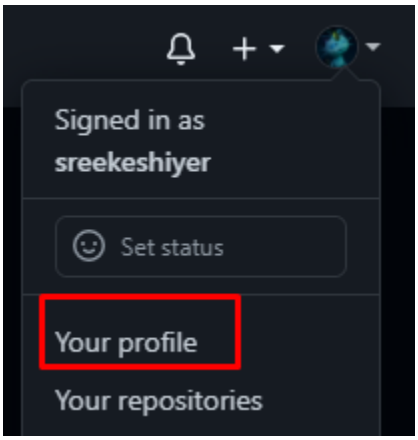
a. Displays the state of the working directory and the staged snapshot.
b. You'll want to run this in conjunction with git add and git commit to seeing exactly what's being included in the next snapshot.

Just getting started – GitHub

Just using GitHub, please complete each task below. This requires that you have already set up an account.

1. Change your profile picture and name

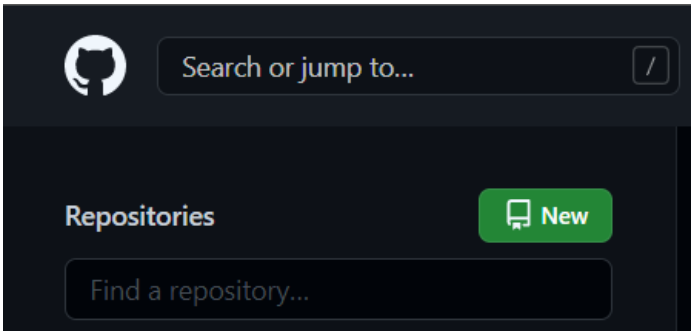
a. After logging in to your GitHub account on github.com, click on profile dropdown at the top right corner and select 'Your Profile'.



b. Click on your default avatar, this will take you to the profiles page from where you can change your profile avatar and add your Name

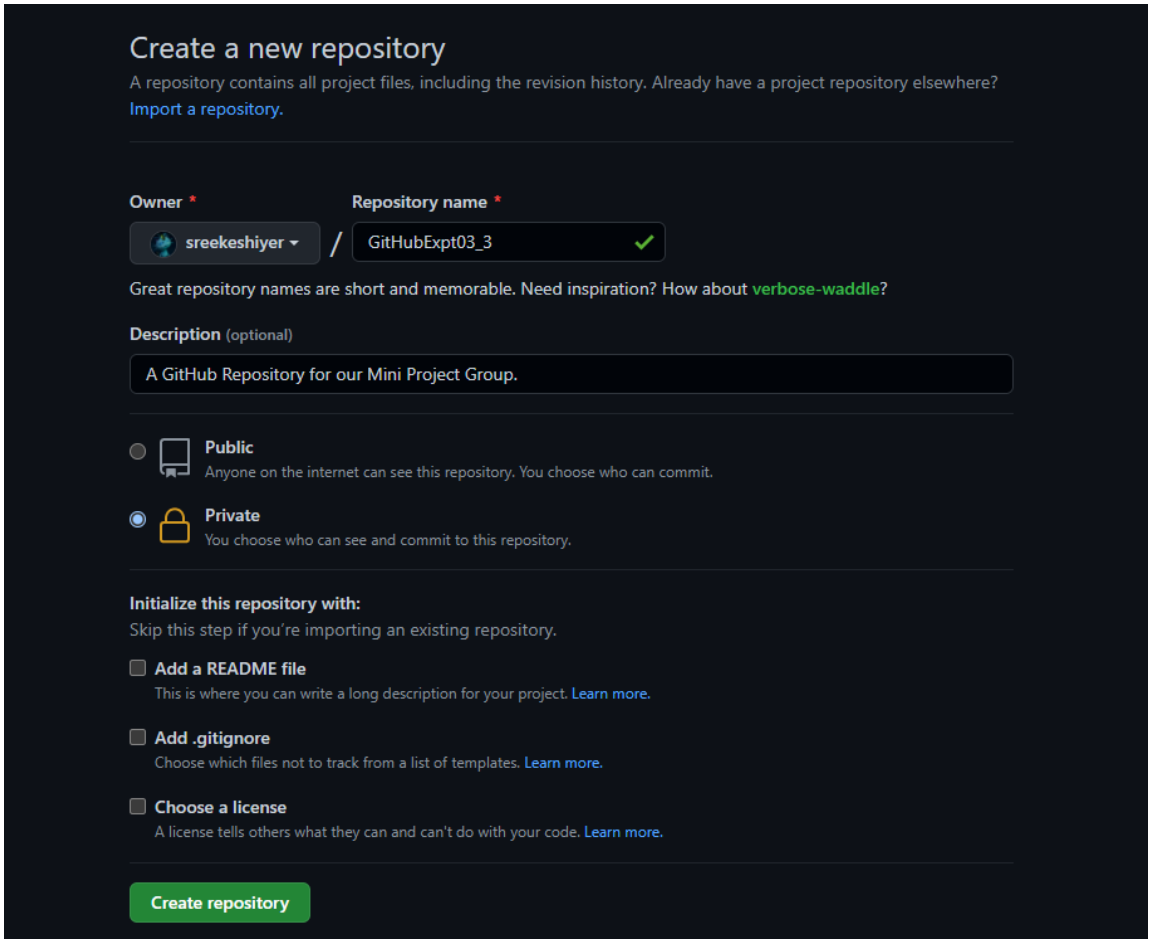
2. Create a repository named GitHubExpt03_grno

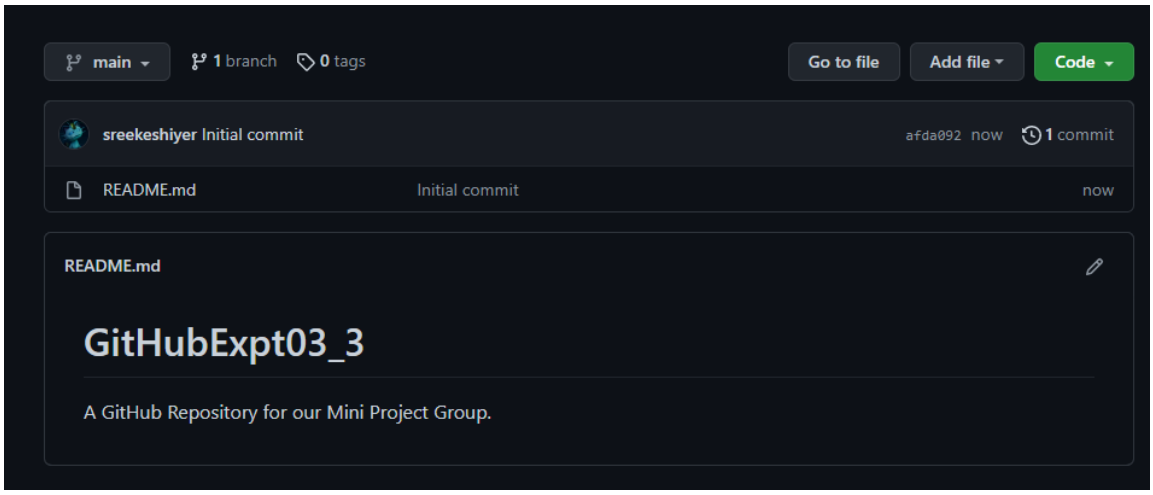
a. In order to create a new repository, navigate to the homepage while logged in to your account.
b. Click on the button 'New'



c. You will be prompted to fill in information like
   i. Name of the repository
   ii. Description
   iii. Visibility of repository (private or public)
   iv. Initialization options

d. Fill in the details as per the requirements and click 'Create repository'

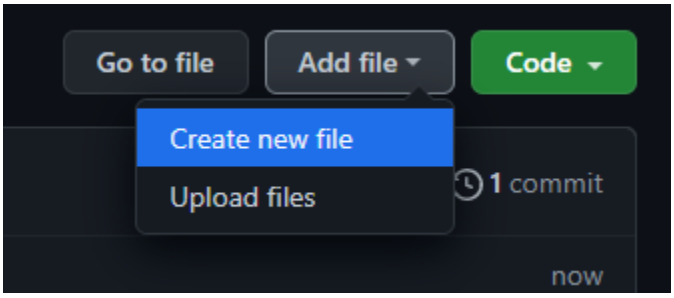




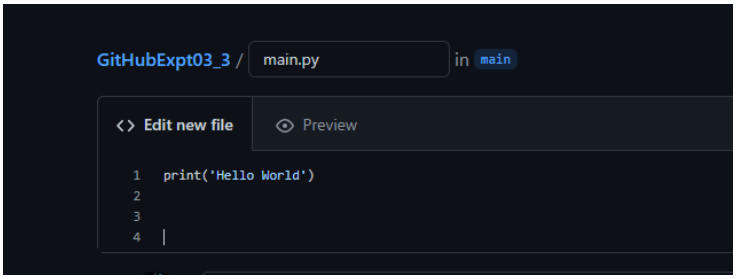
A Repository is created.

3. **Create a file in the above directory named “intro.txt” with your name on the first line.**

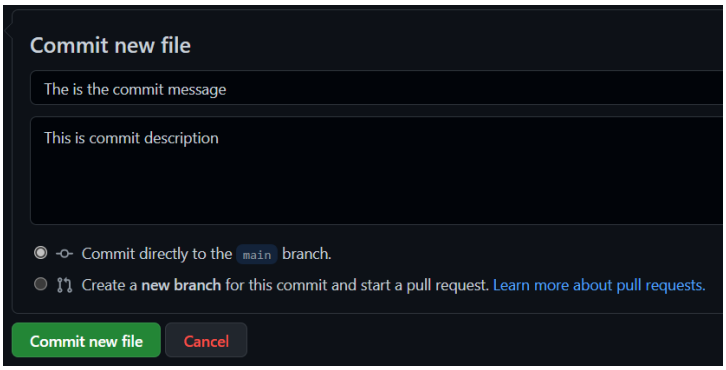
   a. To create a file, click on ‘Add File’ and select 'create new file'



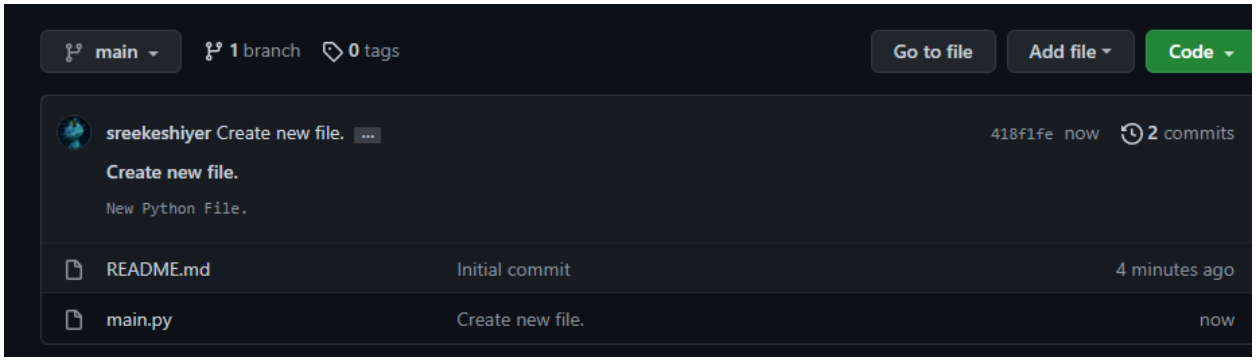
   b. Enter the name of the file, select the branch you want to create the file in (default is main)
   c. Enter the content of the files



d. After adding the content, navigate to the bottom of the page
e. Add commit message and description, select the branch you wish to add the file then click 'Commit new file'



The new file that was created is now seen in the repository with the latest commit and details



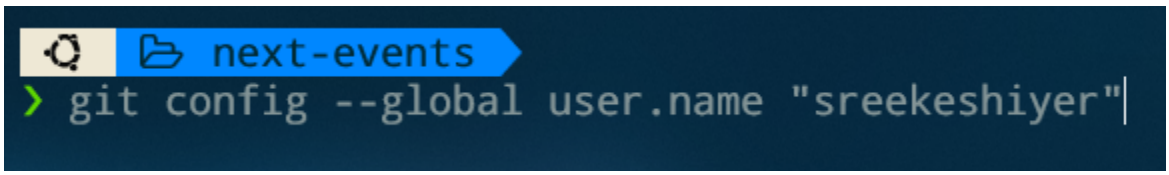
Github repository: https://github.com/Aamir-Ansari-almost/GitHubExpt03_01

***

Just getting started – Git Bash

Just using Git, please complete each task below and write the answer in the blank box beside the question. This requires that you have already installed Git.

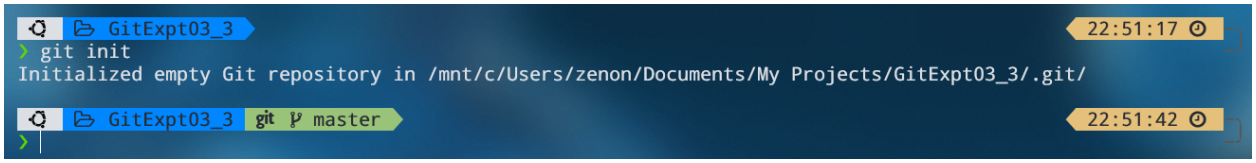
1. **Setup user name**
    a. Open git bash cli
    b. Use the following command to configure your username
        i. *git config --global user.name "Your UserName"*



2. **Setup user email**
    a. Use the following command to configure your email address
        i. git config --global user.email "Your EmailID"

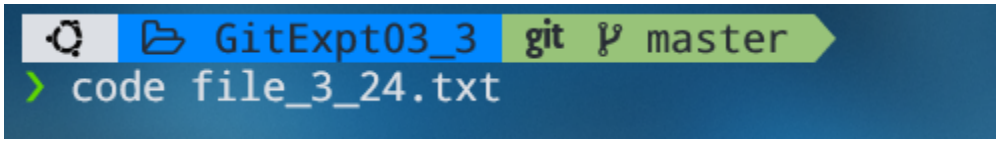
**PS: it is advisable to enter github username and password from windows credential. Since there have been some issue wit github to add username and email address from command line interface*

3. **Create your local repository named GitExpt03_grno**
    a. To initialise a repository use the following command
        i. 'git init'

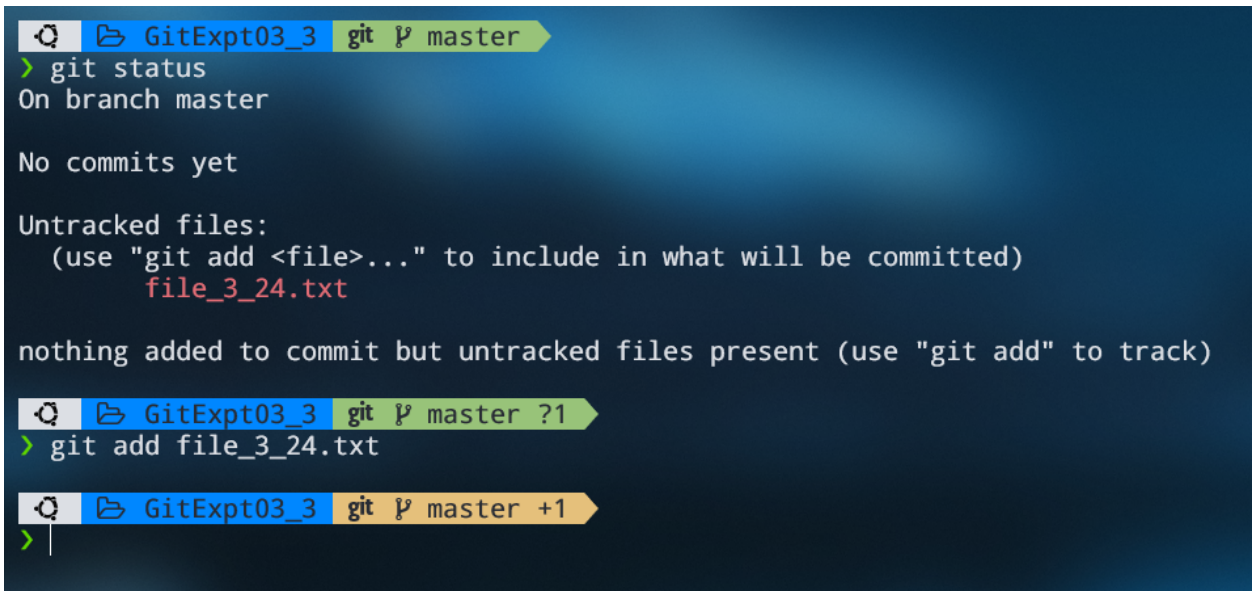


    b. To rename repository, cd into the .git directory and put in the required repository name in the 'description' file

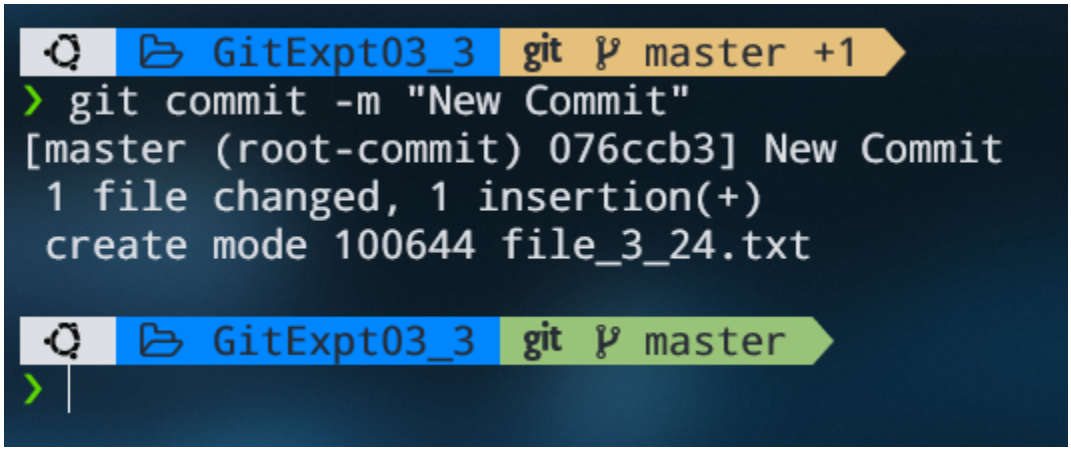
4. **Add a file in above local repository named file_grno_rollno and add some content to the file Add it to the staging state**
    a. Create a file normally with a text editor, and add contents to the file



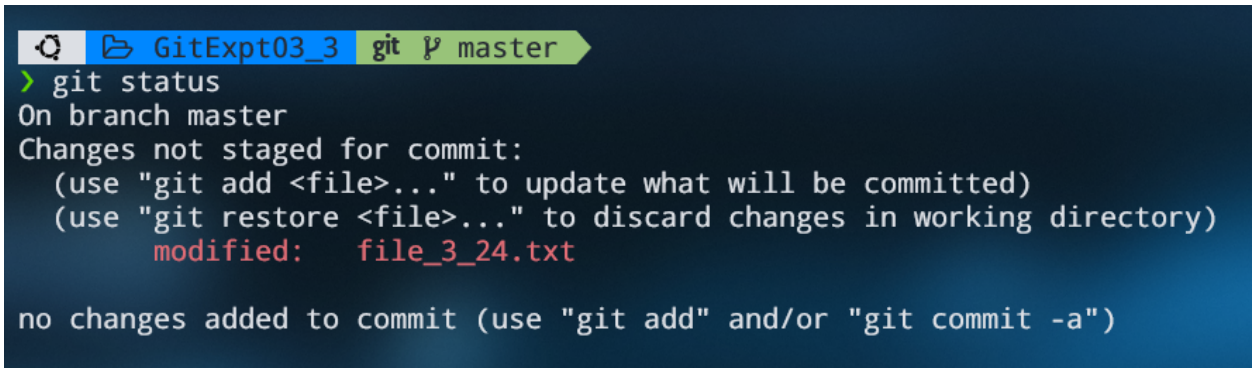
    b. Use the following command to check if some change is made
        i. *'git status'*
    c. Use the following command to add the changes
        i. *'git add <filename>'*



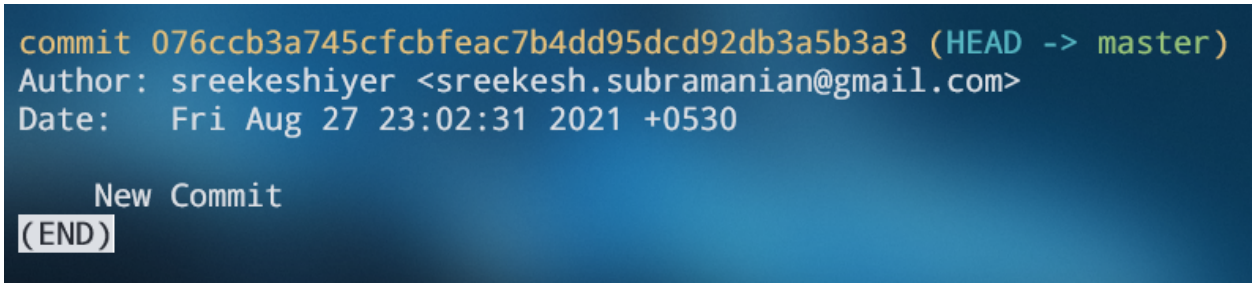
5. **Commit the files**
    a. Use the following command to commit the changes
        i. *'git commit -m "<your message>"'*



6. **Create a scenario for modifications/additions of the multiple files. Show the advantage of staging state.**
    a. If we make some changes at this point in the process, git will show them as an untracked file.
    b. A demonstration is shown below



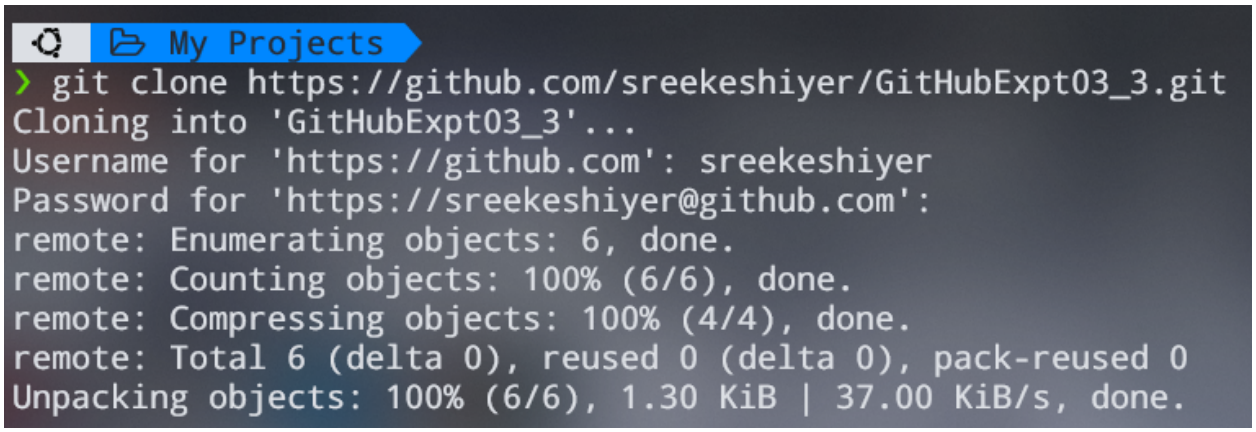
7. **Display the log**
    a. Git log is a utility tool to review and read a history of everything that happens to a repository
    b. Use the following command to view logs
        i. 'git log'



8. Download your GitHub repository created in table-1 named GitHubExpt03_grno

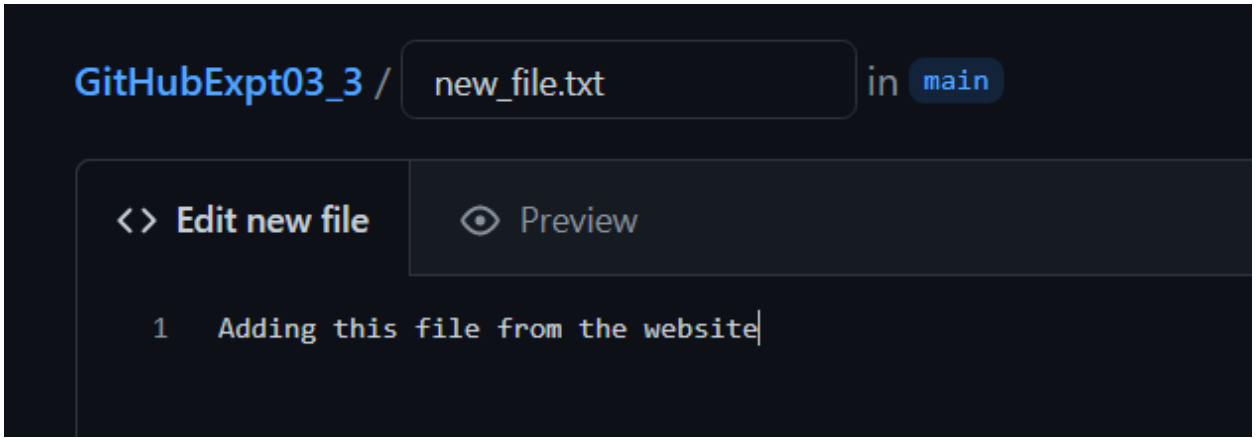
a. To download a repository use the following command

i. 'git clone <URL of repository>'



9. Make some changes in the GitHub file.

a. Add some file directly through github website as shown before



10. Pull the latest files from your GitHub repository

a. Use the following command to check if some changes have been made in the repository and pull those changes

i. 'git fetch'

ii. 'git status'

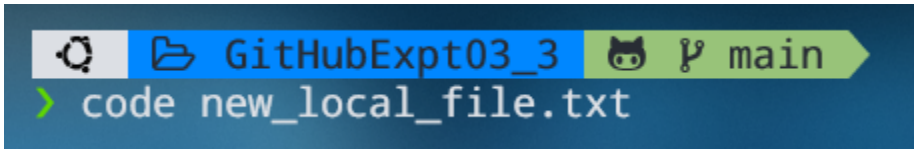
iii. 'git pull'

```
GitHubExpt03_3  main
> git fetch
Username for 'https://github.com': sreekeshiyer
Password for 'https://sreekeshiyer@github.com':
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 730 bytes | 25.00 KiB/s, done.
From https://github.com/sreekeshiyer/GitHubExpt03_3
   418f1fe..8c6b510  main       -> origin/main

GitHubExpt03_3  main ↓1
> git status
On branch main
Your branch is behind 'origin/main' by 1 commit, and can be fast-forwarded.
  (use "git pull" to update your local branch)

nothing to commit, working tree clean

GitHubExpt03_3  main ↓1
> git pull
Username for 'https://github.com': sreekeshiyer
Password for 'https://sreekeshiyer@github.com':
Updating 418f1fe..8c6b510
Fast-forward
 new_file.txt | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 new_file.txt
```

```
GitHubExpt03_3  main
> ls
README.md  main.py  new_file.txt
```

11. Make some changes in the local file of the GitHub repository

a. Make some file normally in your preferred text editor and add content to it
b. You can see that the file is created in the local repository



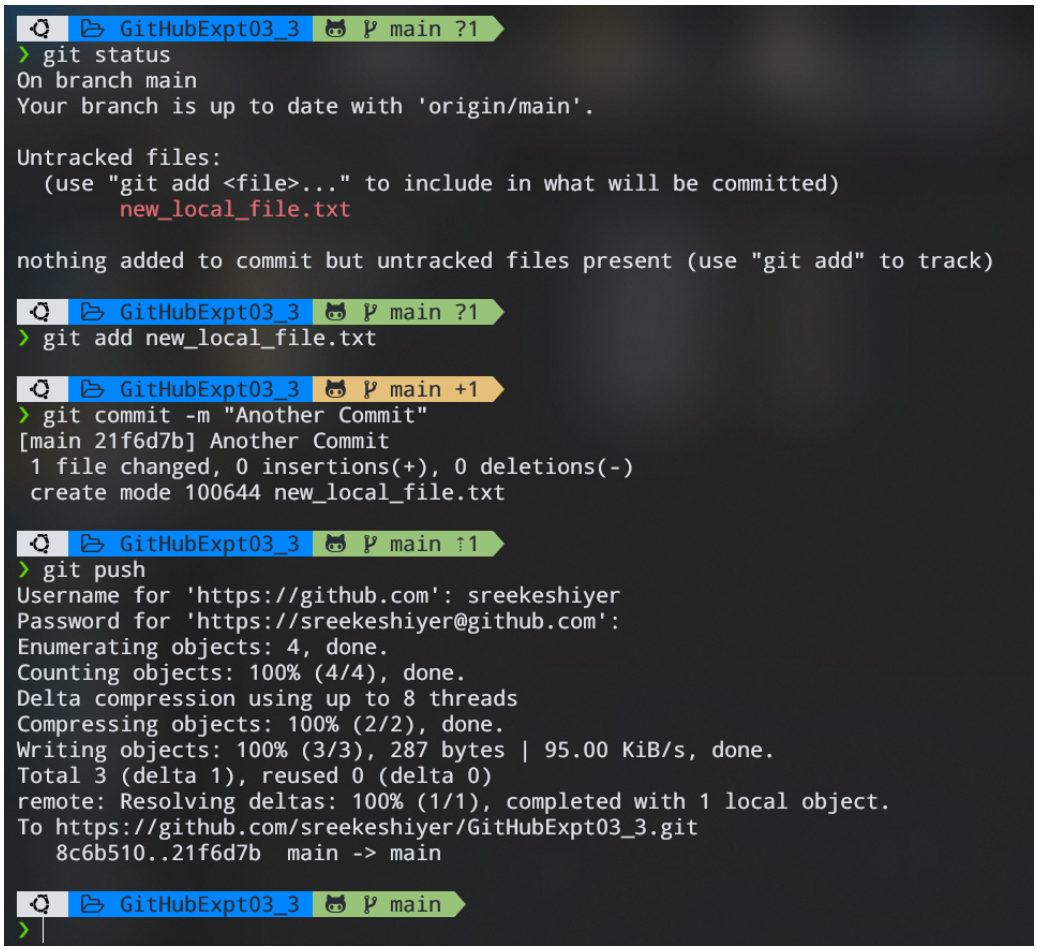
12. Update the changes back to Github Repository

a. Use the following commands to push all local changes to a remote repository

i. *'git add <filename>'*

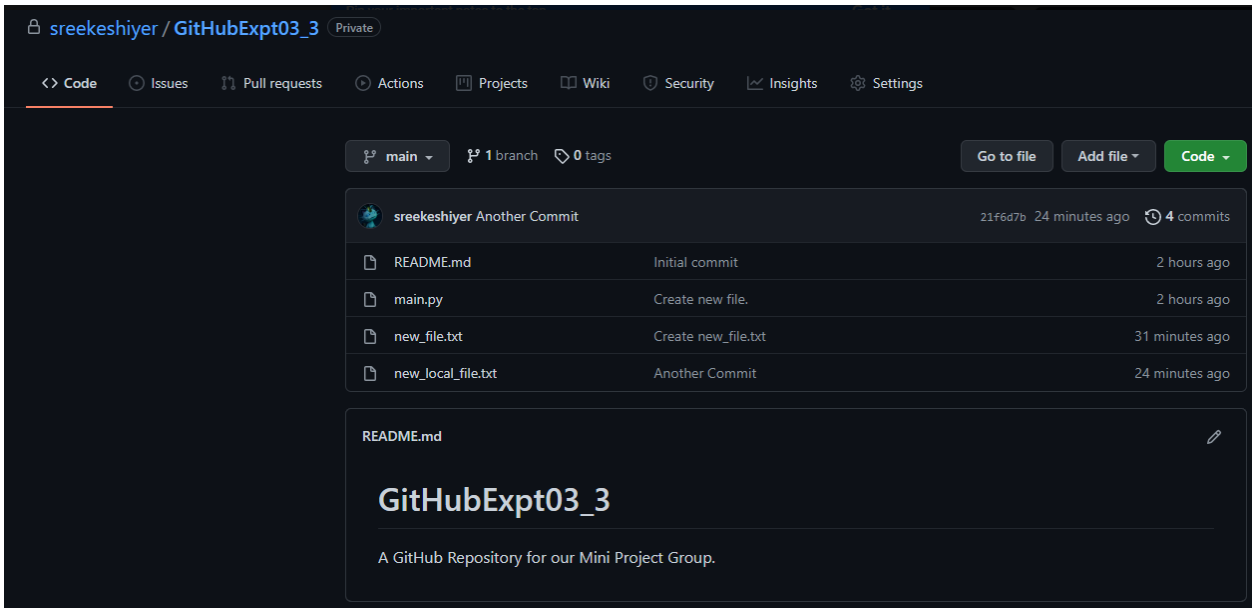
ii. *'git commit -m "<your message>"'*

iii. *'git push'*



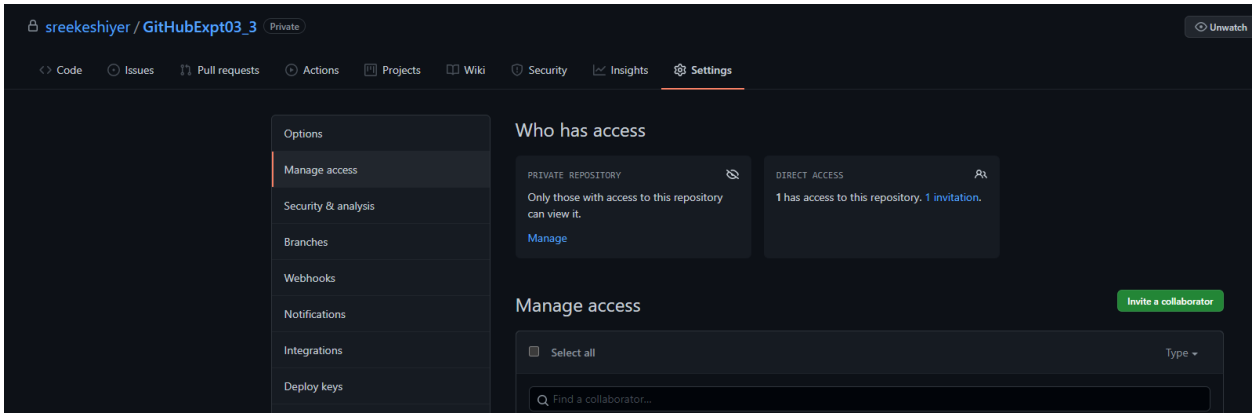
Team set up in GitHub

1. Using one person (“lead developer”) on your team, determine who will house the files in your project.
    a. Create a new repository on GitHub as shown previously
    b. Make sure to set the repository as **Private,** by doing the following we are ensuring that anyone outside our team can not access the repository

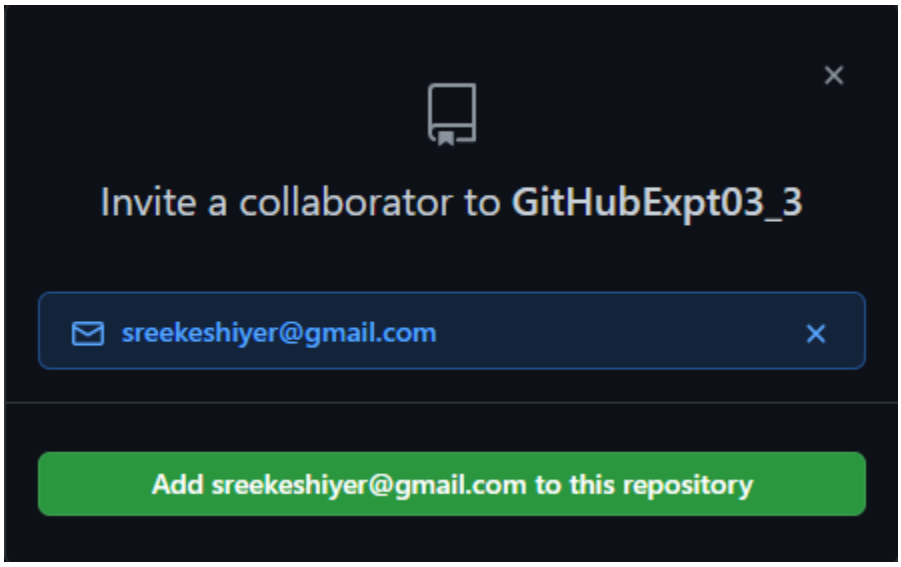


2. That “lead” will create a directory “Project_GrNo” using GitHub.
    a. Now you have created the following repository

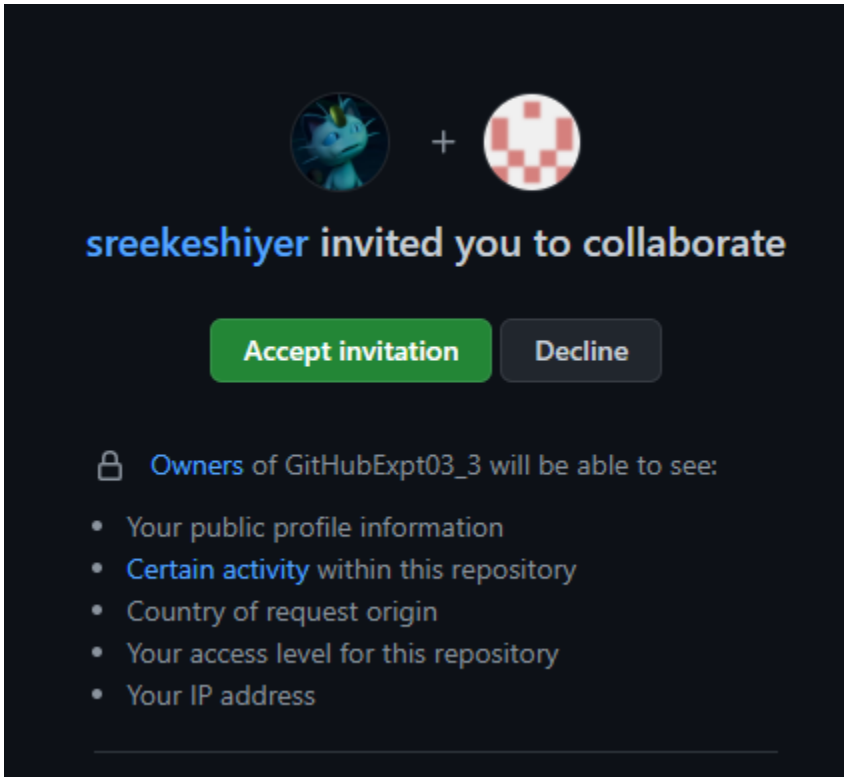
3. Make other team members as collaborators.
    a. Go to setting
        i. Manage access
        ii. Click on the 'Invite a collaborator' button



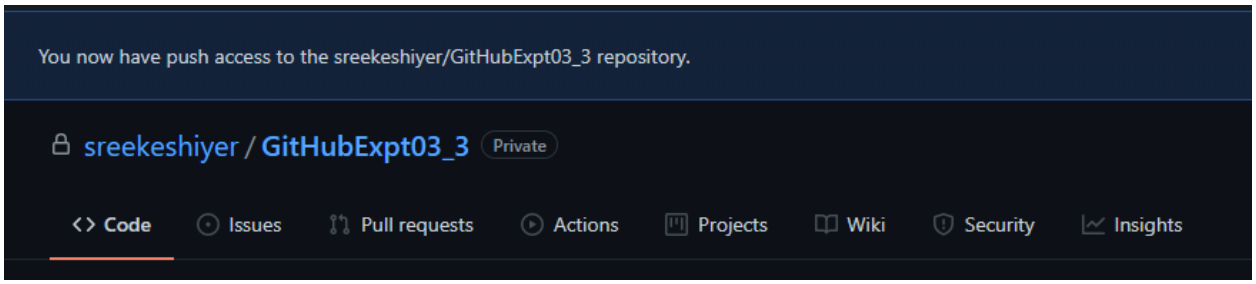
b. Enter GitHub username or email address to add collaborators



4. Using Git, have the others create and add a unique file to the lead's Project_GrNo directory.
    a. The collaborator will get an email and prompt to accept the invitation



b. Newly added collaborators can now access the repository and add files just like shown before



The Lead Developer *sreekeshiyer* added *sreekesh-iyer* to the *Team.* We can see that there is a commit from a collaborator in the repository created by the lead developer.

Conclusion

In this experiment, we learned how to use Git Commands using the Git CLI and we learned how to create repositories, set up teams and collaborate with people on GitHub.