Subject Name: **Source Code Management**

Subject Code: **CS181**

Cluster: **Beta**

Department: **DCSE**

****

|  |  |  |
| --- | --- | --- |
| **Submitted By:**  ABHINAV KUMAR  2110990051  G1 |  | **Submitted To:**  Dr. Monit Kapoor |

***AIM 1: TO SETUP GITHUB CLIENT*.**

TASK 1.1

**THEORY:**

*Git is basically used for pushing and pulling of code. We can use git and git-hub parallelly to work with multiple members or individually. We can make , edit , recreate ,copy or download any code on git hub using git.*

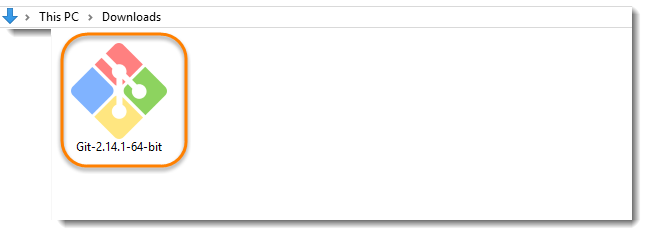
*It’s a Version Control System which means we are able to track all the previous changes in the code using* this software.

**PROCEDURE:**

**STEP1**: *We can install Git on Windows, using the most official build which is available for download on the GIT’s official website or by just typing (scmgit) on any search engine.*

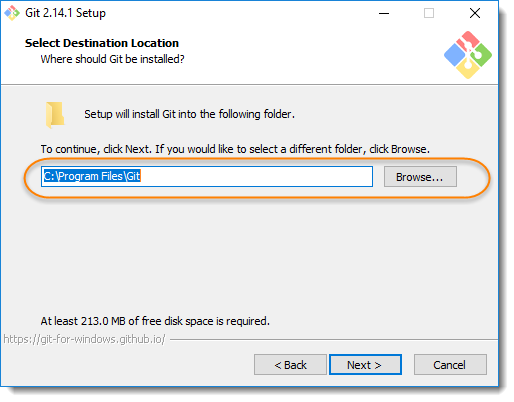
****

**STEP2**: *Click on download for windows. Go to the folder where new downloads gets store, at my machine by default folder is Download folder. Double click on the installer. The installer gets save on the machine as per the Windows OS configuration.*

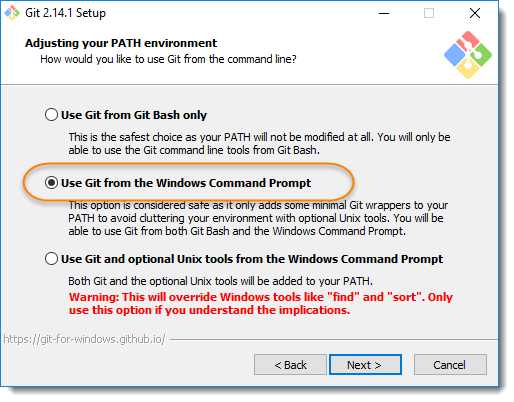


**STEP 3*:*** *When you've successfully started the installer, you should see the Git Setup wizard screen. Follow the Next and Finish prompts to complete the installation.*

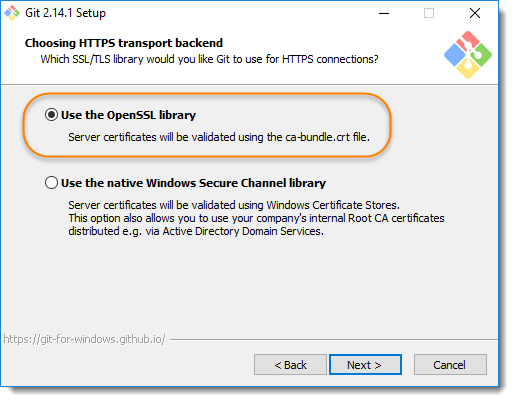
**STEP 4**: *You may like to keep the installation to another folder, so you can set the path of destination location using browse option or either let it be as shown and continue.*



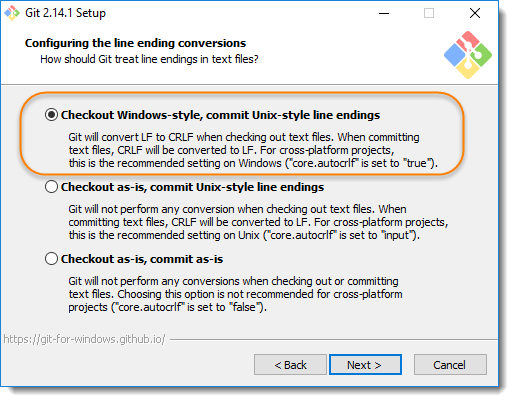
**STEP 5**: *This is asking your choice that whether you like to Git from the****Windows Command Prompt****or you like to use some other program like****Git Bash***.



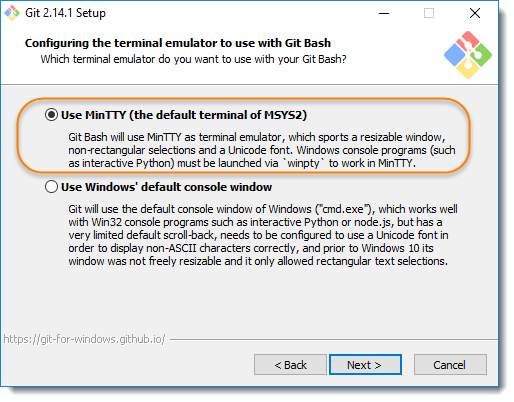
**STEP 6:** Select use the openSSL library



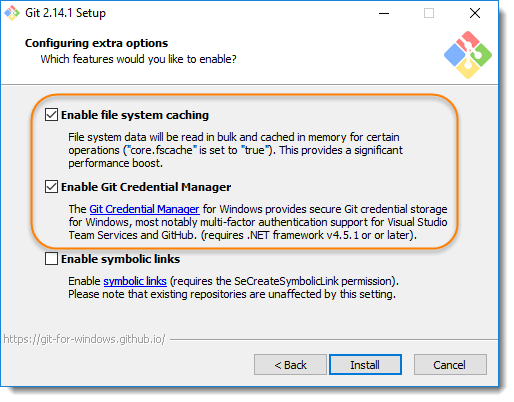
**STEP 7: *select the option and click on next.***



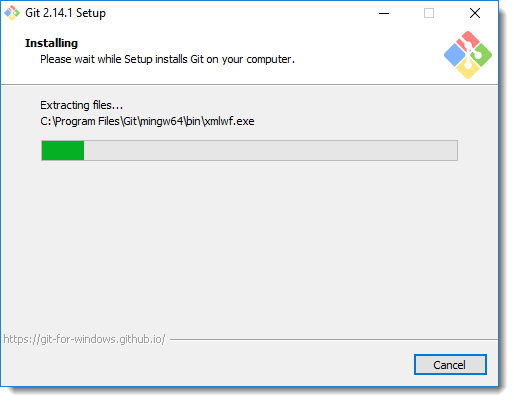
***STEP 8: select the option and click on next.***



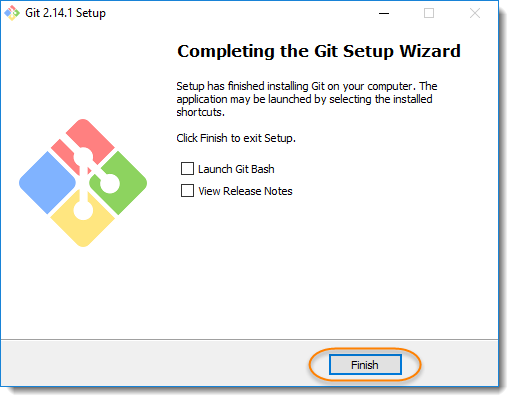
**STEP 9: *Just go with the default selections and click on install***



**STEP 10: Wait *for minutes for the installation to complete.***



**STEP 10: *Once done just click on finish button and the GitHub is installed on your systems.***



TASK 1.2

**AIM 2: TO SETUP GITHUB ACCOUNT**

**THEORY:**

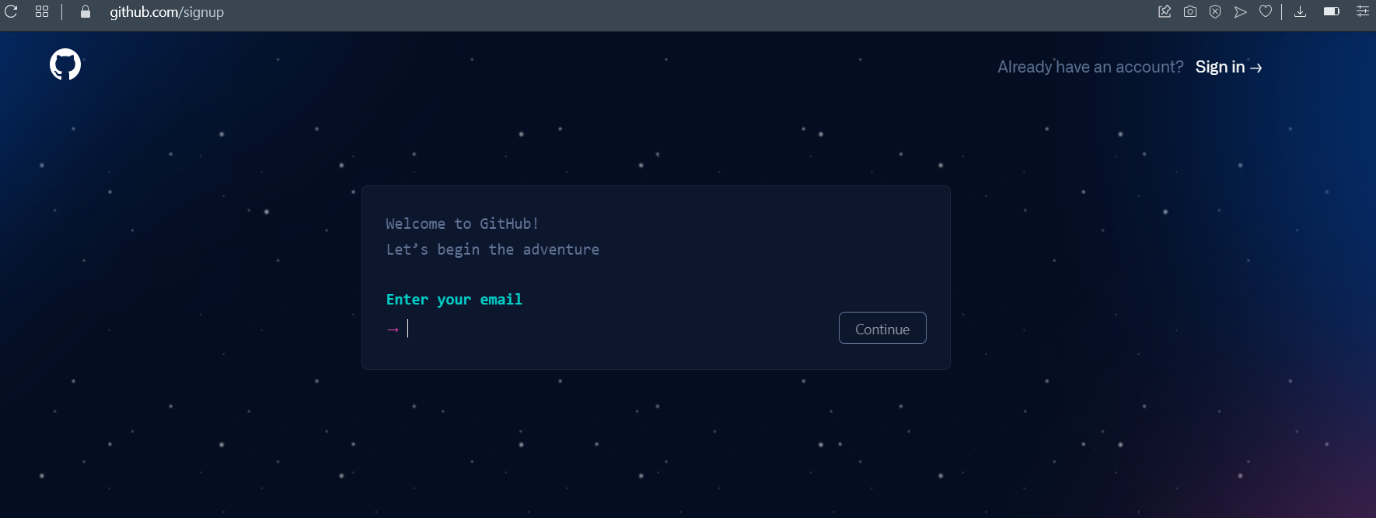
*GitHub is a website and cloud-based service (client) that helps an individual or a developer to store and manage their code. We can also track as well as control* changes to our or public code. GitHub’s has a user-friendly interface and is easy to use .

*We can connect the git-hub and git but using some commands shown below in figure 001. Without GitHub we cannot use Git because it generally requires a host and if we are working for a project, we need to share it will our team members, which can only be done by making a repository . Additionally , anyone can sign up and host a public code repository for free, which makes GitHub especially popular with open-source projects*

**PROCEDURE:**

**STEP1: *On any search engine like google, Microsoft edge, opera, etc. , search for git-hub.***

***A dialogue box would appear welcoming you to the git-***hub.

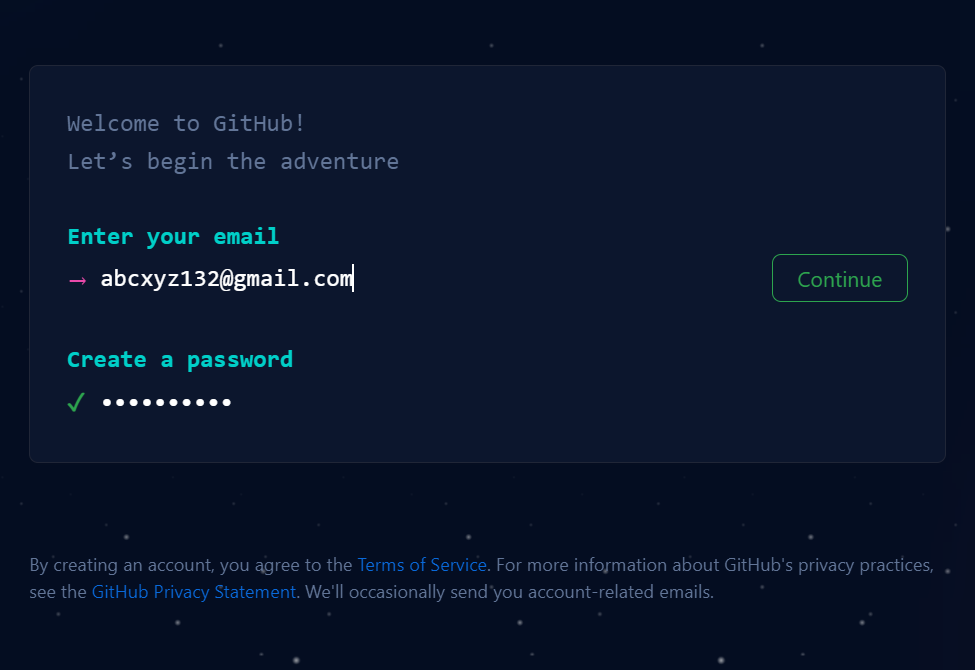


**STEP2: Sign up or log in**

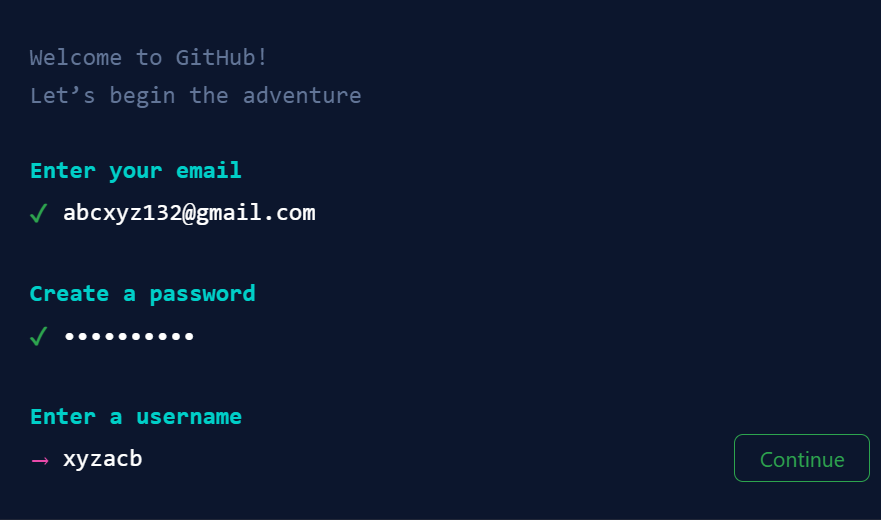
***If you already have an account, then on the top right corner there’s an option of signing up. Click on “SIGN IN “.***

**STEP 3: *But if you don’t have any account, then enter your email and continue***

**STEP 4: *Then create a strong password for your GitHub account***



**STEP 4:** ***Create a username***

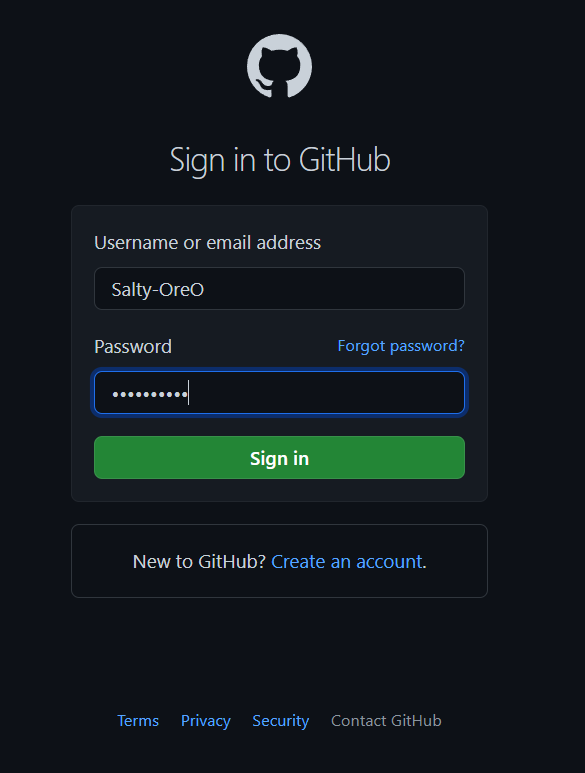
****

**STEP 4: *fill in all the details that are required and click on create account and your GitHub account would be created***

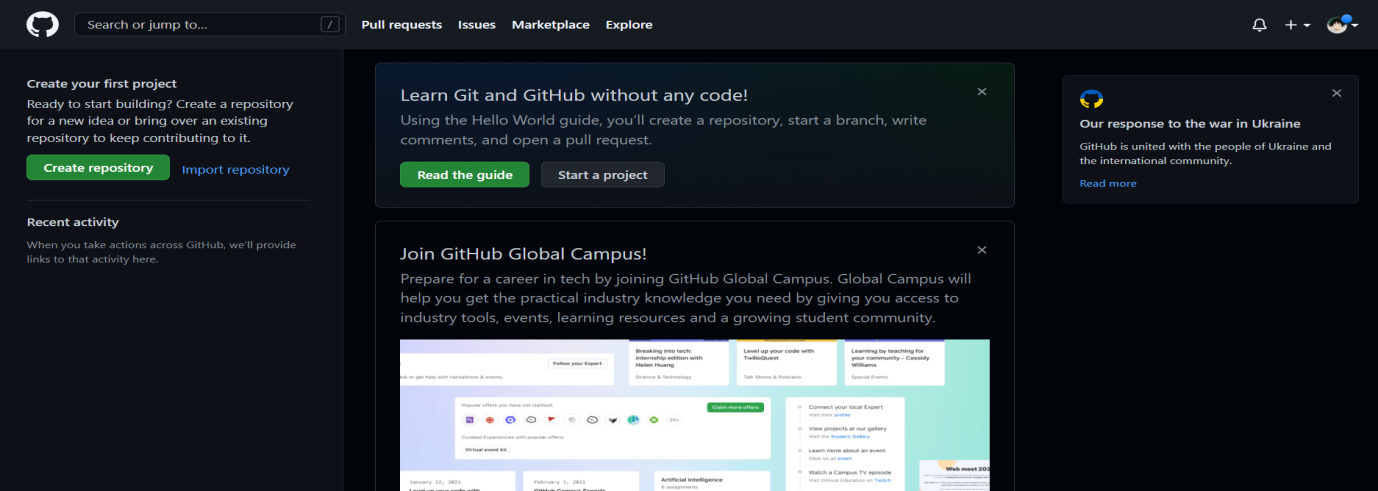
BUT IF YOU ALREADY HAVE AN ACCOUNT, THEN FOLLOW THESE STEPS:

**STEP 1: Sign In**

***Enter your email-address or username , then the password and continue***



**STEP 2: *you have logged in in your account. Now you can create and edit any project***



TASK 1.3

**AIM 3: PROGRAM TO GENERATE LOGS**

**THEORY:**

*Logs are nothing but the history which we can see in git . It contains all the past commits, insertions and deletions in it which we can see any time.*

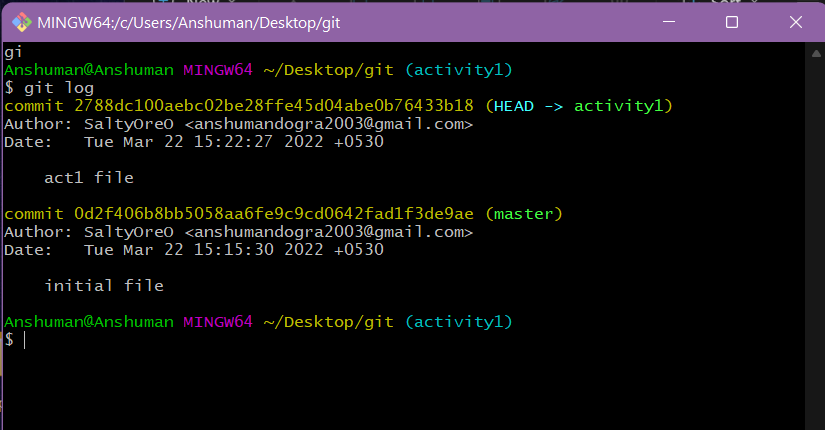
*Logs helps to check that what were the changes in the code or any other file and by whom. It also contains the number of insertions and deletions including at which time it was changed.*

**PROCEDURE:**

**The command used to generate logs in git is :**

**$ git log**

***$ git log***



* As it can be observed, on using this command, the system displays all the changes made in the file or list of all the commits in the history along with the information of the user.
* **This commands clearly defines the git as the ‘version-controlled system’ as it allows us to rollback to any of the previous working states and keeps track of all the versions.**

TASK 1.4

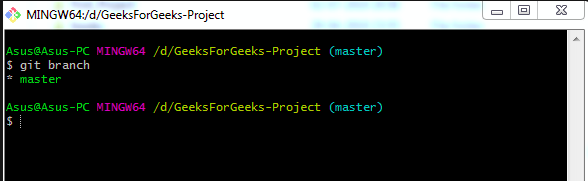
**AIM 4: TO CREATE AND VISUALIZE BRANCHES.**

**THEORY:** The main branch in git is called as master branch. But we can make branches out of this main master branch. All the files present in master can be shown in branch but the file which are created in branch are not shown in master branch. We can also merge both the parent (master) and child (other branches).

FOLLOW THESE STEPS TO CREATE A SEPARATE BRANCH IN GIT.

1. **Firstly, you can check the present branches in the master branch with the help of this command.**

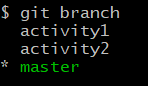
$ git branch



1. **To create a branch, enter:**

**$ git branch \_\_\_\_\_**

**Write the name of the branch you want to create**

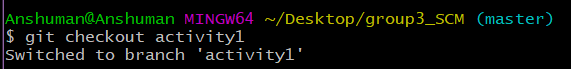


* The branch has been created.

1. **The next step is to transfer the data from the master branch to the new branch. For this we use:**

**$ git checkout \_\_\_\_\_\_\_**

**Name of the new branch that was created**

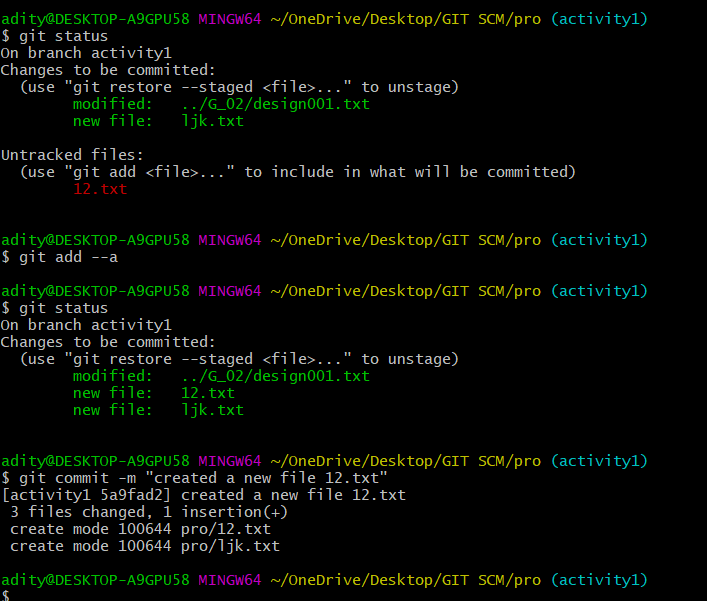


* All the data from the master branch has been transferred to this branch**.**

1. **Staging the file.**

Check the status of the file.

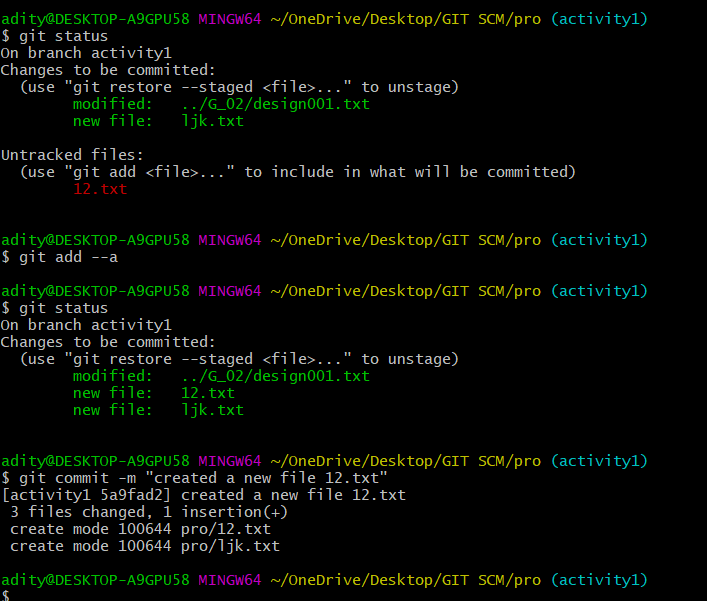
You would observe the name of the file is in red colour with a notation “untracked “.



This means that only the data has been transferred to the file but we cannot make changes in the same as the current working directory is the master branch.

To overcome this, we stage the file by using the command **:**

**$ git add –a**



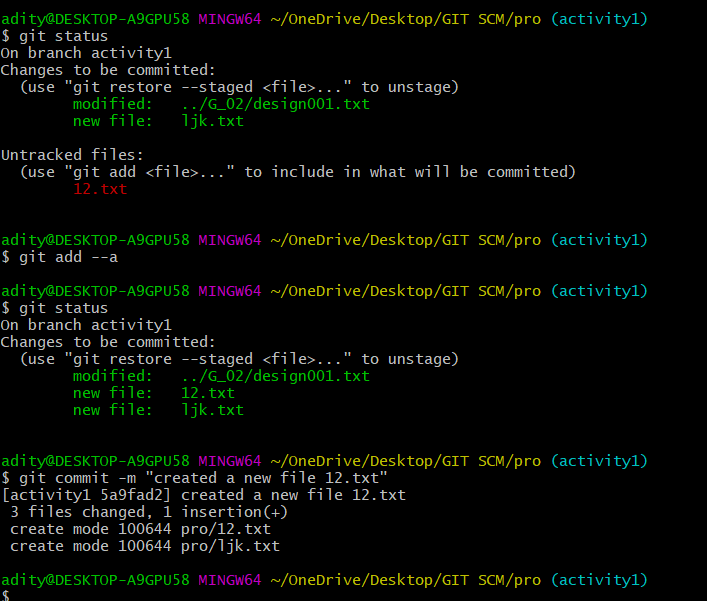
Check if the file is staged or not by using $ git status command.

* As you can observe the file name is green in colour now which denotes the file is staged.

1. **Commit**

After staging we need to commit. This assures the system that the directory has been shifted to the new file. For this, the command used is:

**$ git commit-m \_\_\_\_\_\_\_\_**



1. **Check the status**

On checking the status, a message will be displayed as ‘ working tree clean’. Which means all the files inside that directory are tracked.

Now you can make any of the changes in the file but the modifications won’t be reflected in the master branch.

TASK 1.5

**AIM5: TO EXPLAIN GIT LIFECYCLE**

**THEORY:** When a project is under Git version control system, they are present in three major Git states in addition to these basic ones. Here are the three Git states**:**

1. **Working directory**
2. **Staging area**
3. **Git directory**

***1.Working Directory***

Consider a project residing in your local system. This project may or may not be tracked by Git. In either case, this project directory is called your Working directory. Working directory is the directory containing hidden .git folder***.***

***2.Staging area***

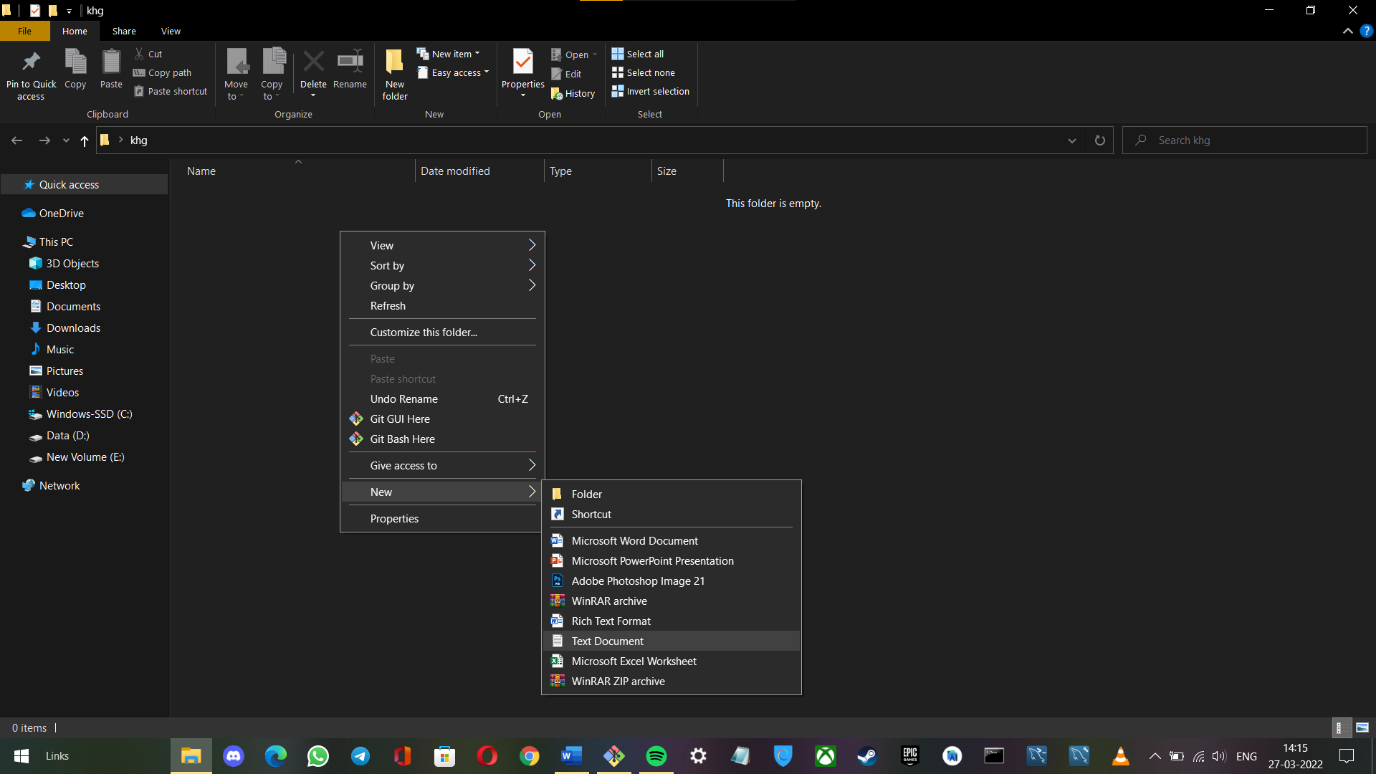
Staging area is the playground where you group, add and organize the files to be committed to Git for tracking their versions**.**

***3.Git Directory***

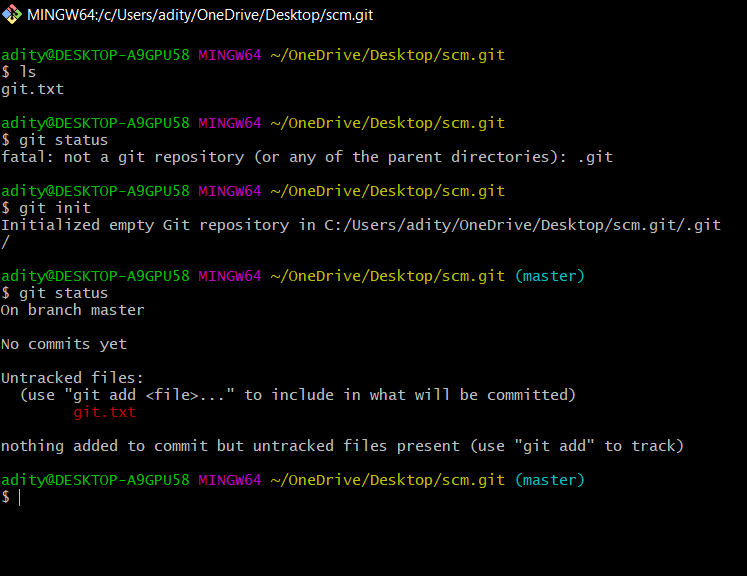
Now that the files to be committed are grouped and ready in the staging area, we can commit these files. So, we commit this group of files along with a commit message explaining what is the commit about. Apart from commit message, this step also records the author and time of the commit. Now, a snapshot of the files in the commit is recorded by Git. The information related to this commit is stored in the Git directory.

**PROCEDUE: Suppose you want to add a file, follow these steps.**

**STEP1: Create a file in the folder.**

****

**STEP 2: check status**

****

It will show the file name in red colour. This means that the file is untracked.

**STEP 3: Tracking the working directory**

**Use the command :**

**$ git init**

This command would help the git to track this directory. Or we can say, that we’ve created a git repository in this existing project directory.

**STEP 4: Staging**

In this step, we would be selecting the files that have to be tracked by git. That's because some files in the project like class files, log files, result files and temporary data files are dynamically generated and there’s no need to track these versions.

Whereas the source code files, data files, configuration files and other project artifacts contain the business logic of the application. These files are to be tracked by Git are thus needs to be added to the staging area.

Command: **$ git add –a**

****

**STEP 5: Check the status**

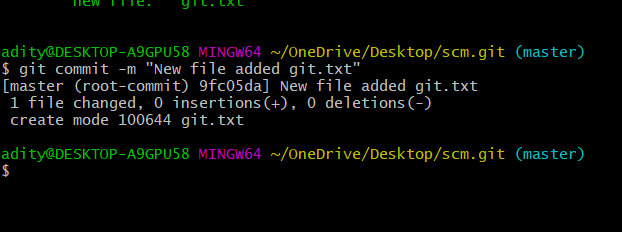
Check if the file is staged or not by using $ git status command.

****As you can observe the file name is green in colour now which denotes the file is staged.

**STEP 6: Committing to git repository**

Now that the files are ready in the staging area, we can commit these files.  So, we commit this group of files along with a commit message explaining what is the commit about.

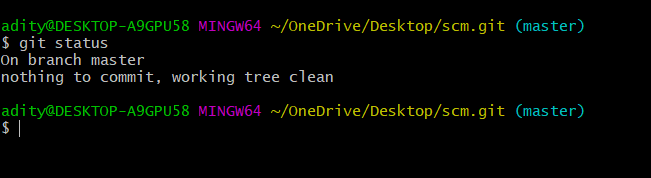
Commit files to git repository with message

****

* Now, a snapshot of the files in the commit is recorded by Git. The information related to this commit (*names of files committed, date and time of commit, author of commit, commit message*) is stored in the Git directory.

**STEP 7: Check the status again.**

This time a message will be displayed as ‘working tree clean’. Which means all the files inside that directory are tracked.

****