

**Experiment No. 01**

**Aim:** Setting up of Git Client

**Theory:**

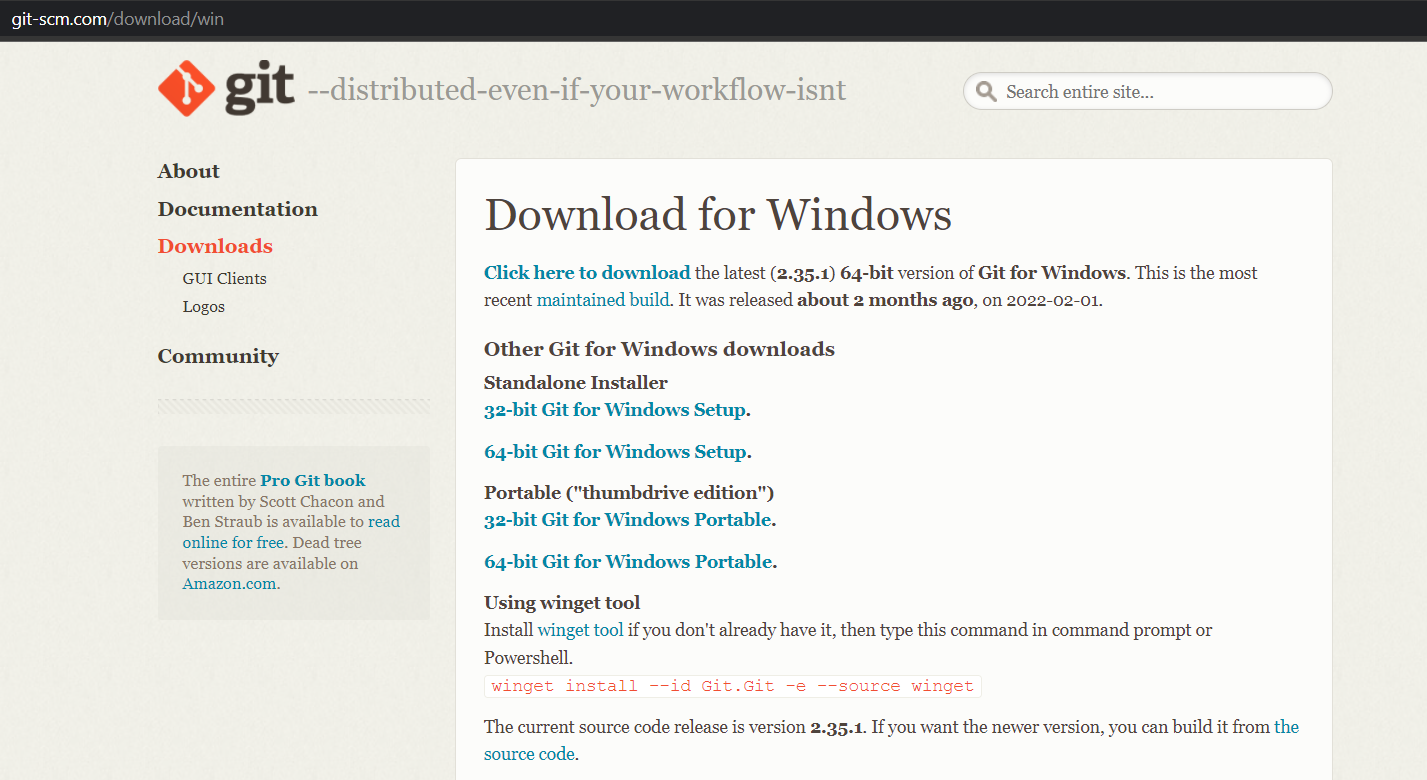
GIT : It is basically used for pushing and pulling of code. We can use git and git-hub parallelly to work with multiple members or individually. We can make , edit , recreate ,copy or download any code on git hub using git.

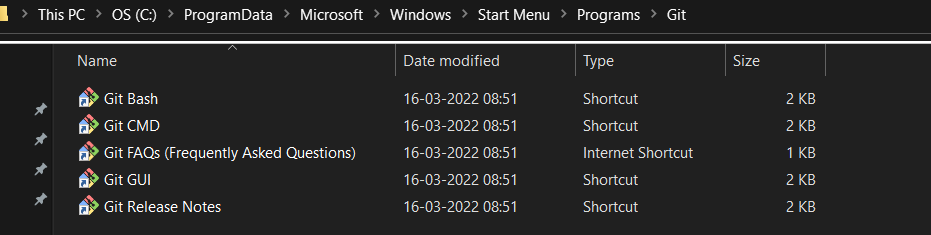
What is GIT ? : It’s a Version Control System(VCS) -> It is a software or we can say a server by which we are able to track all the previous changes in the code.

Advantages of GIT :

**Procedure:** 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 ( s c m git ) on any search engine . We can go on <https://git-scm.com/download/win> and can select the platform and bit-version to download. And after clicking on your desired bit-version or ios it will start downloading automatically.

**Snapshots of download:**

****





**Experiment No. 02**

**Aim:** Setting up GitHub Account

**Theory:**

**What is GitHub ->** GitHub is a website and cloud-based service (client) that helps an individual or a developers to store and manage their code. We can also track as well as control changes to our or public code.

**Advantages of GitHub ->** 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:-**

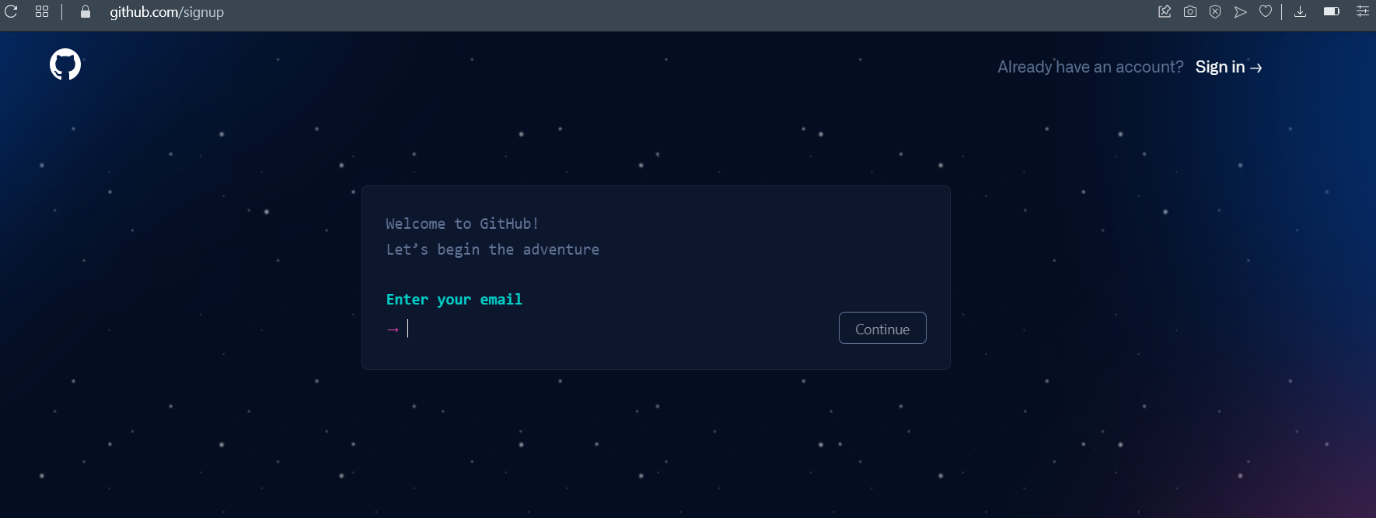
**Step1 :-**

Google (any search engine)

Search for git-hub or (<https://github.com/signup>).

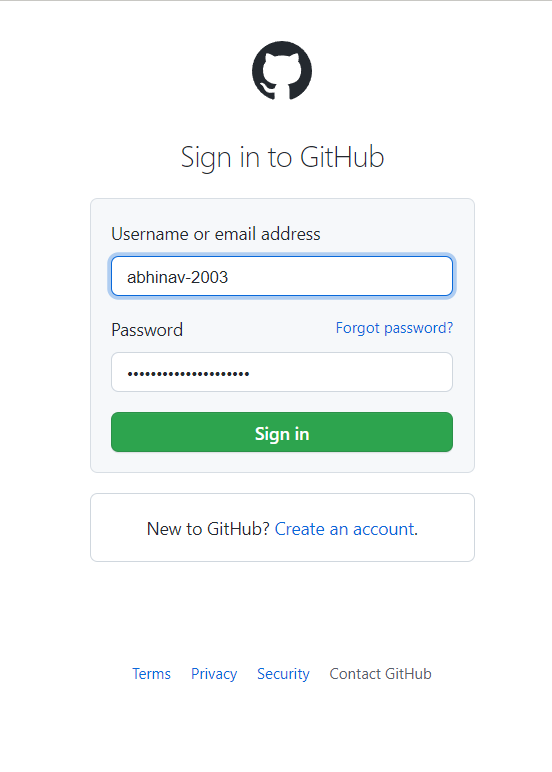
**Step2 :-**

**Snapshots** –

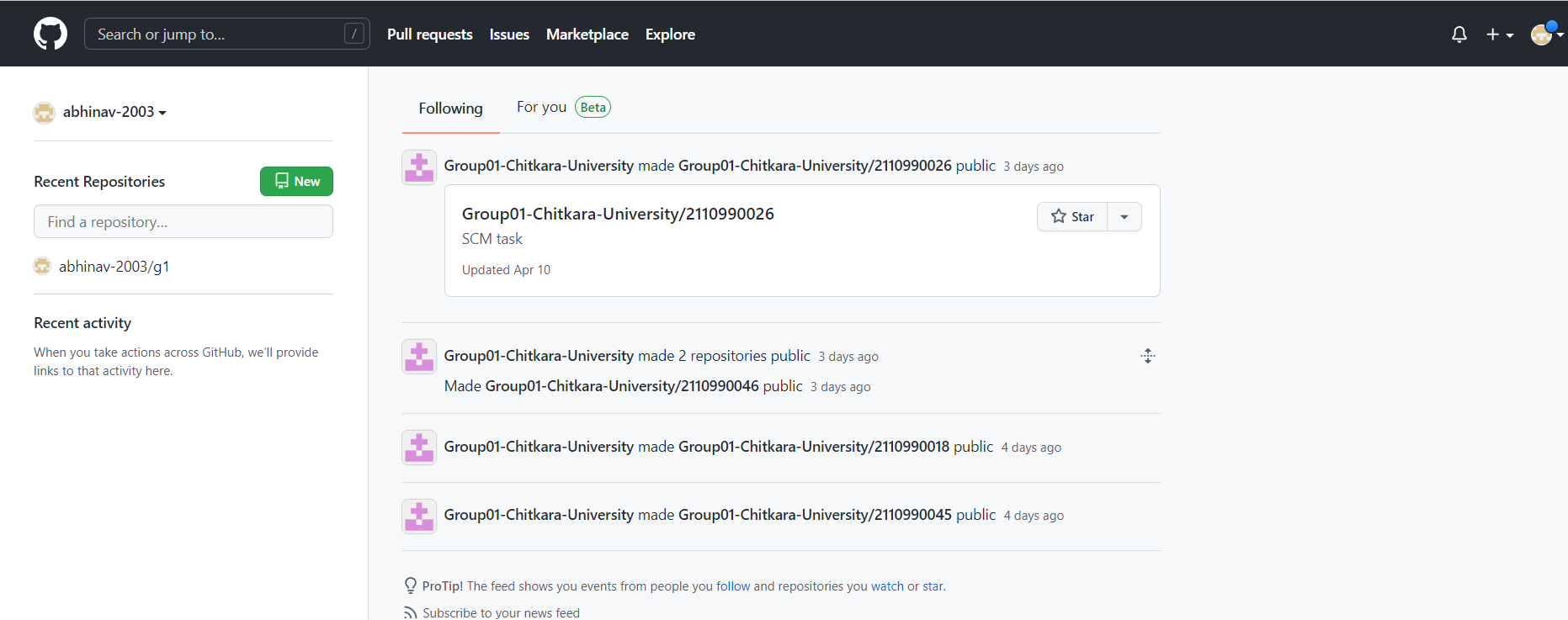


After visiting the link this type of interface will appear, if you already have account you can sign in and if not you can create.

**Sign in into GIT-HUB :-**



**Interface of GitHub :-**

gir

**To link GitHub account with Git bash –**

**For username:-**

git config --global user.name “username in git-hub”

**For user email:-**

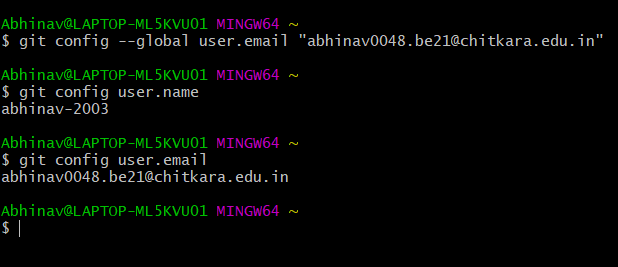
git config --global user.email “your email in git-hub”

**To verify:-**

git config user.name

git config user.email

**Snapshot :-**

****



**Experiment No. 03**

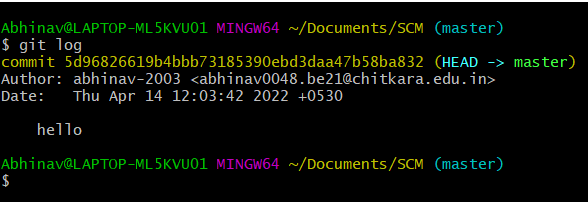
**Aim:** Program to Generate log

**Theory:-**

**Logs ->** Logs are nothing but the history which we can see in git by using the code git log.

It contains all the past commits, insertions and deletions in it which we can see any time.

**Why logs ->**  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.





**Experiment No. 04**

**Aim:** Create and visualize branches

**Create branches :-**

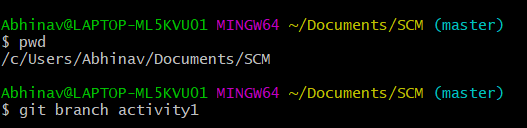
The main branch in git is called as master branch. But we can make branches out of this main master branch. All the files present in master can be shown in branch but the file which are created in branch are not shown in master branch. We can also merge both the parent (master) and child (other branches).

**Syntax:-**

1. For creating a new branch.

git branch name of branch

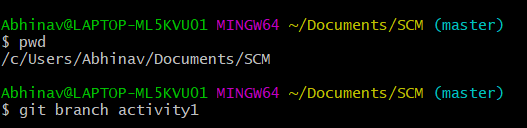
**Snapshots –**



1. We can also check how many branches we have.

git branch

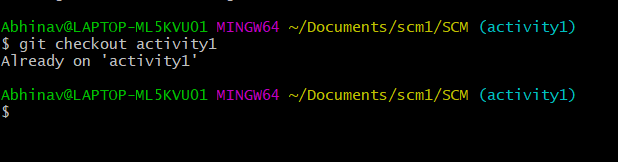
**Snapshots :-**



1. To change the present working branch.

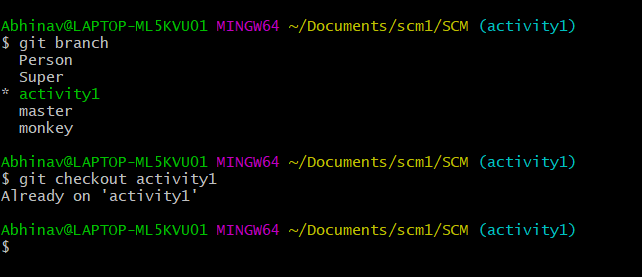
git checkout name of branch.

**Snapshots –**

****

**Visualizing branches :-**

To visualize I have created a new file in a new branch activity 1 instead of master branch.



After this I have done the 3 step architecture which is tracking the file , send it to stagging

area and finally we can role back to any previously saved version of this file.

In this way we can create and change different branches . We can also merge the branches

by using git merge command.



**Experiment No. 05**

**Aim:** Git lifecycle description

**Theory:**

**Stages in GIT Life Cycle ->** Files in a Git project have various stages like Creation, Modification, Refactoring, and Deletion and so on. Irrespective of whether this project is tracked by Git or not, these phases are still prevalent. However, 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:

* Working directory
* Staging area
* Git directory

**Working Directory ->**

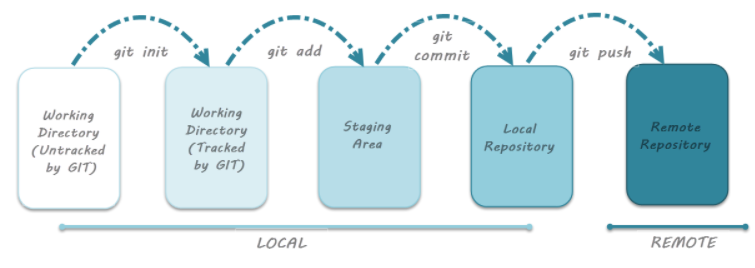
Consider a project residing in your local system. This project may or may not be tracked by Git. In either case, this project directory is called your Working directory.

### **Staging Area ->**

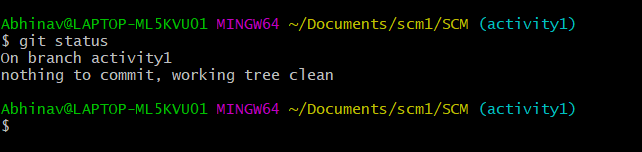
**Staging area is the playground where you group, add and organize the files to be committed to Git for tracking their versions.**

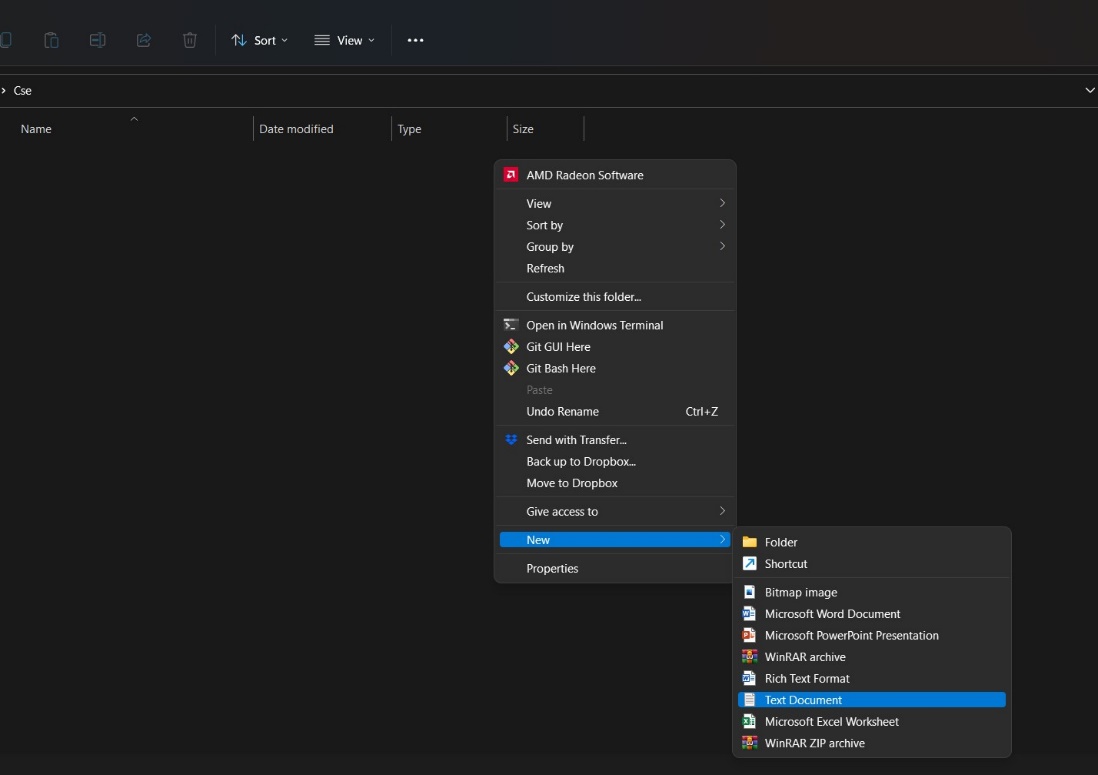
**Git Directory ->**

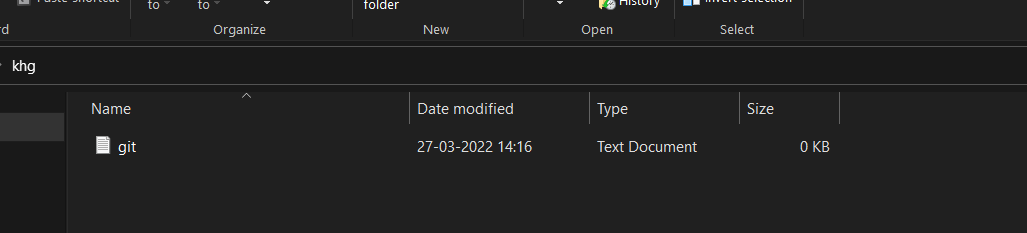
Now that the files to be committed are grouped and ready in the staging area, we can commit these files. So, we commit this group of files along with a commit message explaining what is the commit about. Apart from commit message, this step also records the author and time of the commit. Now, a snapshot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

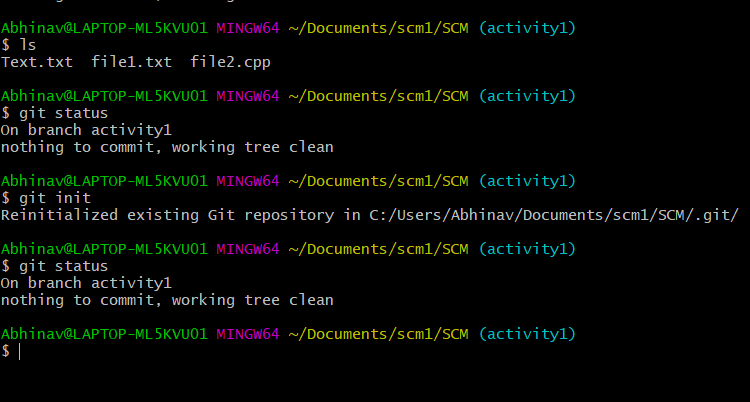
**Remote Repository-> means mirror or clone of the local Git repository in GitHub**. And **pushing means uploading the commits from local Git repository to remote repository hosted in GitHub.**

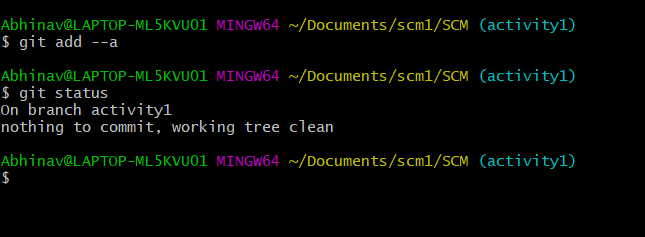
**Snapshots –**

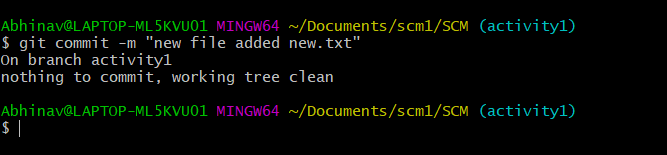
****

****

****

****

****

****